NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

THERMINATOR: CONFIGURING THE UNDERLYING STATISTICAL MECHANICS MODEL

by

Daniel W. Ettlich

December 2003

Thesis Co-Advisors: John C. McEachen
Chris S. Eagle

Approved for public release; distribution is unlimited

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE			Form Approved OMB No. 0704-0188
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instruction, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) Washington DC 20503.			
1. AGENCY USE ONLY *(Leave blank)*	**2. REPORT DATE** December 2003	**3. REPORT TYPE AND DATES COVERED** Master's Thesis	
4. TITLE AND SUBTITLE: Therminator: Configuring the Underlying Statistical Mechanics Model			**5. FUNDING NUMBERS**
6. AUTHOR(S) Ettlich, Daniel W.			
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000			**8. PERFORMING ORGANIZATION REPORT NUMBER**
9. SPONSORING /MONITORING AGENCY NAME(S) AND ADDRESS(ES) N/A			**10. SPONSORING/MONITORING AGENCY REPORT NUMBER**
11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.			
12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release; distribution is unlimited			**12b. DISTRIBUTION CODE**

13. ABSTRACT ***(maximum 200 words)***

The rapid increase in sophisticated Internet attacks has left the security industry lagging far behind. In an attempt to improve network security, Therminator, a patternless intrusion detection system, was developed in 2001 by NPS in conjunction with NSA. The Therminator model uses statistical mechanics to analyze network traffic as a system of exchanges. Being highly configurable enables Therminator to be adapted for any network configuration. Until now, however, no exploration had been conducted on the configuration parameters of the underlying statistical mechanics model. It is important to understand the effects of these parameters to optimize anomaly detection. Thus the current study explored these parameters using HTTP traffic generated in a controlled test environment. Results were as follows: equations were developed for state counting to determine bucket state space sizes; bucket state space size was found to be symmetrical about the midpoint of the boundary conditions; proper display period was based on traffic rate; and lastly, the more orthogonal anomalous traffic was to the normal traffic, the larger the perturbation was in the state graph. These results provide needed insight into properly configuring Therminator for optimal anomaly detection, ultimately affording the Department of Defense greater network security.

14. SUBJECT TERMS Network Security, Network Assurance, Information Protection, Intrusion Detection, Patternless Intrusion Detection, Network Anomaly Detection, Real-Time Network Monitoring, Statistical Mechanics			**15. NUMBER OF PAGES** 97
			16. PRICE CODE
17. SECURITY CLASSIFICATION OF REPORT Unclassified	**18. SECURITY CLASSIFICATION OF THIS PAGE** Unclassified	**19. SECURITY CLASSIFICATION OF ABSTRACT** Unclassified	**20. LIMITATION OF ABSTRACT** UL

NSN 7540-01-280-5500 Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release; distribution is unlimited

THERMINATOR: CONFIGURING THE UNDERLYING STATISTICAL MECHANICS MODEL

Daniel W. Ettlich
Lieutenant, United States Navy
B.S/B.A., University of San Diego, 1994
MBA, The University of Arizona, 2001

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN ELECTRICAL ENGINEERING
and
MASTER OF SCIENCE IN COMPUTER SCIENCE

from the

NAVAL POSTGRADUATE SCHOOL
December 2003

Author: Daniel W. Ettlich

Approved by: John C. McEachen
Thesis Advisor

Chris S. Eagle
Co-Advisor

John P. Powers
Chairman, Department of Electrical and Computer Engineering

Peter J. Denning
Chairman, Department of Computer Science

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

The rapid increase in sophisticated Internet attacks has left the security industry lagging far behind. In an attempt to improve network security, Therminator, a patternless intrusion detection system, was developed in 2001 by NPS in conjunction with NSA. The Therminator model uses statistical mechanics to analyze network traffic as a system of exchanges. Being highly configurable enables Therminator to be adapted for any network configuration. Until now, however, no exploration had been conducted on the configuration parameters of the underlying statistical mechanics model. It is important to understand the effects of these parameters to optimize anomaly detection. Thus the current study explored these parameters using HTTP traffic generated in a controlled test environment. Results were as follows: equations were developed for state counting to determine bucket state space sizes; bucket state space size was found to be symmetrical about the midpoint of the boundary conditions; proper display period was based on traffic rate; and lastly, the more orthogonal anomalous traffic was to the normal traffic, the larger the perturbation was in the state graph. These results provide needed insight into properly configuring Therminator for optimal anomaly detection, ultimately affording the Department of Defense greater network security.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

LIST OF FIGURES

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF TABLES

THIS PAGE INTENTIONALLY LEFT BLANK

ACKNOWLEDGMENTS

I would first like to thank my wife Jenna for all of her support during the past two years. Without her help and understanding regarding my late hours at work and thesis travel, this would have been a much more difficult ordeal. I would like to thank Dr. John McEachen for his guidance and providing me with the resources necessary to conduct this study as effectively as possible. I want to thank CDR Chris Eagle for taking the time out of his very busy schedule to assist me on this project. Last, I want to thank the Ther-minator support group consisting of Stephen Donald, Robert McMillen, Dr. Dave Ford, Steve Huntsman, and Dr. John Zachery.

THIS PAGE INTENTIONALLY LEFT BLANK

EXECUTIVE SUMMARY

The proliferation of the Internet into society has brought with it unprecedented capabilities for information sharing, albeit at a significant cost. The Internet now connects multinational governments, commercial, private, and educational networks as an infrastructure of autonomous systems. Originally developed as an educational tool, connectivity and ease of use were emphasized over security, which has created a haven for abuse. From script kiddies to state sponsored professional hackers, this lack of robust security results in annual damages of hundreds of millions of dollars. The rapid increase in sophisticated attacks has left the security industry lagging far behind. In an attempt to improve network security, Therminator, a patternless intrusion detection system, was developed in 2001 by NPS in conjunction with NSA. The Therminator model uses statistical mechanics to analyze network traffic as a system of exchanges. This model is highly configurable, enabling Therminator to be adapted for any network configuration.

Therminator provides a graphical representation of the network traffic flow. Anomalous activity appears as perturbations in the Therminator GUI. The GUI consists of two major parts. The upper graphic, dubbed the *Thermal Towers*, represents the average ball count in each bucket for each display period. The lower graphic, dubbed the *Thermal Canyon*, represents the frequency of bucket states for each display period.

Until now, however, no exploration had been conducted on the configuration parameters of the underlying statistical mechanics model. It is important to understand the effects of these parameters to optimize anomaly detection. The main thrust of the study was to explore these configuration parameters to gain needed insight into properly configuring Therminator for optimal anomaly detection, ultimately affording the Department of Defense greater network security.

The current study explored these parameters using hypertext transfer protocol (HTTP) traffic generated in a controlled test environment. A test network was developed around the Spirent Smartbits TeraMetrics traffic generator. The system was capable of generating line speed traffic. The traffic scheme used consisted of simple eight packet HTTP exchanges. With all the traffic flowing in and out of the network, the traffic rate can be controlled by varying the number of users per second.

The first part of this study involved defining the bucket state space. Equations were then developed for state counting to determine the size of bucket state space. The size of a bucket state space given a set of initial and boundary conditions was determined with

$$N = \binom{n + n(i-l) - 1}{n-1} - \sum_{j=1}^{\lfloor n(i-l)/(u-l+1) \rfloor} (-1)^{j+1} \binom{n}{j} \binom{n + (n*i) - j(u+1) - \sum_{j+1}^{n} l - 1}{n-1},$$

where n is the total number of buckets, i is the total number of balls, u is the upper limit of balls in a given bucket, and l is the lower limit of balls in a given bucket.

This equation is necessary in understanding how to properly configure the Therminator. Next, the concept of a *smoothing ratio* was developed to define the averaging time for data within the Therminator statistical mechanics algorithm. It was determined that a low smoothing ratio results in increased false positives and a high ratio results in increased false negatives.

In exploring the SLIDELENGTH, it was determined that the display period, which is equal to the SLIDELENGTH, should be set based on the traffic rate. A relationship between the display period and traffic rate was not able to be empirically determined, but was found to be less than linear. It was also determined that optimizing a system for a reduced traffic rate would require a less than linear increase in the SLIDELENGTH. The WINDOWLENGTH would also increase, but not as much as the SLIDELENGTH. This results in a decrease in the smoothing ratio.

It was determined that the SLIDELENGTH and WINDOWLENGTH settings affect the bucket space as a system. Not all systems will have even traffic distribution across the bucket space. For situations with uneven traffic distributions, optimization can be achieved by changing the BUCKETSPACEINIT parameters. The smallest grouping within a bucket space was determined to be a conversation group. Uneven traffic distributions within a conversation group could not be optimized by changing the BUCKETSPACEINIT parameters. In this case, a bucket prioritization scheme must be used. For uneven traffic distribution between conversation groups, optimization could be

achieved by changing the BUCKETSPACEINIT parameters. In this case, non-homogenous boundary conditions could optimize the system.

In experiments conducted on varying the BUCKETSPACEINIT parameters, it was determined that reducing the size of the bucket state space had a similar effect as increasing the traffic rate. In addition, the size of the bucket state space was symmetrical about the midpoint of the boundary conditions, assuming homogeneity across the bucket space with respect to the BUCKETSPACEINIT parameters. Last, a translation property was discovered with the BUCKETPSACEINIT parameters. Two BUCKETSPACEINIT parameter sets were equivalent if one was a scalar translation of the other.

Lastly, the idea of orthogonal traffic was developed. The ability to detect anomalous traffic was determined to be based on the quantity of packets and their orientation to the normal traffic flow. The more anomalous packets received and/or the more orthogonal the traffic is to the normal traffic, the larger the perturbation in the Therminator GUI and thus a greater chance of being detected.

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF SYMBOLS, ACRONYMS, AND/OR ABBREVIATIONS

3D	3 Dimensional
BC	Boundary Condition
DoS	Denial of Service
GUI	Graphical User Interface
HTTP	Hypertext Transfer Protocol
IC	Initial Condition
IDS	Intrusion Detection System
IP	Internet Protocol
k	Thousand
ms	Milliseconds
NOC	Network Operations Center
NPS	Naval Postgraduate School
NSA	National Security Agency
PERL	Practical Extraction and Report Language
pps	Packets per Second
SR	Smoothing Ratio
SL	SLIDELENGTH
TCP	Transport Control Protocol
WL	WINDOWLENGTH

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

A. MOTIVATION

The development of a secure and dependable networked computer infrastructure is dependent upon real-time detection of anomalous events and behaviors. These events and behaviors are typically propagated over a network from source host to victim host. Today, most networks are linked together forming a system dubbed the Internet. The Internet was originally intended to be a tool for research and education, not an international matrix of commercial, private, government, and educational systems. With this in mind, the protocols governing its operation were originally designed for flexibility and connectivity, not security [1].

Several approaches to intrusion detection have been studied and applied in practice with varying effects. The earliest intrusion detection systems (IDS) integrated signature-based analysis for detection with normal network models [2]. Since then, many different systems have been based on the same assumption that malicious network activity is inherently different than normal network activity. This is especially true of signature-based IDS, which compare real events to known malicious or abnormal events. The weakness of these types of systems is poor detection of new attacks, variations of known attacks and attacks that are similar to normal activity. Attacks with explicit syntactic signatures are easily detectable, but many attacks are much more subtle. In addition, ambiguities in network protocols can lead to attacks that subvert signature-based IDS and launch undetectable insertion and evasion attacks [3, 4].

As attackers have become stealthier and more savvy, the incidents and scope of denial or service (DoS) attacks has increased dramatically. This includes widespread attacks from network worms. A worm is a small program that infects a host computer, replicates itself, and then exploits the host's network connection to infect other hosts [5]. This results in two major problems, network denial of service from the exponential replication and the possibility of a malicious payload infecting the attacked hosts. In 2001, the Code Red worm propagated to over 359,000 Internet hosts in less than 14 hours [6]. Two years later, the slammer worm propagated to over 75,000 hosts in less than 30 min-

utes, 90 percent of these in the first 10 minutes [7]. The scope of DoS resulting from worm attacks is quick and widespread, even global, in nature. The Computer Security Institute's 2003 Computer Crime and Security Survey reported that 82% of respondents had been a victim of a virus attack and 42% a DoS attack in the last year. Reported losses for both types of attacks reached nearly $100 million [8].

Current network security mechanisms are inadequate for handling the threat posed by a worm attack. It has been argued that automated containment of self-propagating code is more likely to be successful than prevention or treatment mechanisms [9]. Successful containment mechanisms require early and rapid detection of network and host threats. This type of approach emphasizes process rather than user monitoring. Mitigating propagation speed of worm attacks depends on accurate and efficient monitoring of host-based and network-based events for anomalous behaviors.

The Therminator model formally characterizes the *conversations* between a large number of interacting entities in a computer network. Capturing the distributions of conversation flow rates and sizes between network nodes is fundamental to the model. The primary problem for early network attack detection is efficient data reduction that accurately and rapidly distinguishes between normal and anomalous activity in the presence of an overwhelming amount of data related to network conversation flow. The Therminator's data reduction and anomaly detection model surpasses aggregate statistical models to provide a holistic view of conversation exchange dynamics and anomalous deviations from the normal baseline. This holistic view of network conversation rates and flows supports early containment strategies for worms by providing indication of anomalous events leading to DoS. The Therminator model provides rapid early detection and holistic network monitoring capabilities in a single model together with an effective visualization process.

B. HISTORY

The Therminator model spawned out of a directive issued by the National Security Agency to solve the network security problem. The model used by Therminator was the brainchild of Dr. Dave Ford who developed it while working for NSA. The original version of Therminator (version 1), dubbed Thermonator was written in PERL. The program was tested at the Pacific Command's NOC in early 2001. During these tests, Thermonator was evaluated by a team from NSA. The evaluation concluded Thermonator would not detect typical attacks with the exception of attacks resulting in significant changes in the traffic flow [10]. This first version of Therminator had rudimentary graphs displaying the energy and entropy associated with the network traffic flow. The notion of state was not integrated into this version, thus hampering the ability to detect anomalous activity.

In the spring of 2001, Dr. Ford requested the assistance of Dr. John McEachen at the Naval Postgraduate School. Dr. McEachen sent two students, LT Steve Donald and Capt. Robert McMillen, to assist Dr. Ford with the project. Fifty-two days later, a completely redesigned version of Therminator (version 2) was completed. This version was coded in the C programming language and included a third-party 3D graphical overlay. To date, version 2 is the base for all later versions of Therminator. This version incorporated the notion of "state", which is essential to the statistical mechanical core model.

Therminator has been through several revisions since version 2, each adding increased capability. The version of Therminator that is used in this thesis is a modified version 3. The core model is identical to version 2, with minor cosmetic and functionality improvements. The Therminator program is infinitely configurable, which results in difficulty configuring it properly. The purpose of this thesis was to explore two major configuration areas, the SLIDELENGTH/WINDOWLENGTH parameters and the BUCKETSPACEINIT parameters.

C. SUMMARY

In the hostile environment of today's networks, monitoring and anomaly detection are necessary mechanisms for ensuring a secure and dependable computing infrastructure. The Therminator Patternless Intrusion Detection System was designed to fill the gap between required security and commercially available IDS. The next chapter of this thesis focuses on explaining the concepts behind the Therminator model, specifically relating to statistical mechanics. The third chapter defines the concept of state as it related to bucket states and bucket state space. This chapter provides equations for calculating the size of the bucket state space given a set of boundary and initial conditions. The fourth chapter explains the setup of the test network used to generate the traffic for each of the experiments. Chapter five explores the configuration parameters of SLIDELENGTH, WINDOWLENGTH, and the smoothing ratio. Numerous experiments were conducted to determine the effects of these parameters on the performance of the system. Chapter six explores the BUCKETSPACEINIT configuration parameters. Again, numerous experiments were conducted to determine the effects of these parameters on the performance of the system. The last part of this chapter relates to optimizing a system with unequal traffic distribution by employing non-homogeneous boundary conditions across the buckets. Chapter seven contains information on constructing BUCKETSPACEs. This information is derived from observation made during the experiments in Chapters five and six. Chapter seven provides a summary of the thesis and suggestions for follow-on work.

II. THERMINATOR OVERVIEW

A. CHAPTER OVERVIEW

This chapter explains the basic concept behind the Therminator. It describes the statistical mechanical model and how it is used to develop useful information about network traffic. The Therminator GUI is explained including the Thermal Towers and Thermal Canyon graphs.

B. CONCEPT

A network is basically a collection of complex interacting systems, with network traffic being the mode of exchange between these systems. The purpose of Therminator is to allow for accurate detection of anomalous network activity through the reduction of standard network data into a meaningful and useful form [11]. This is accomplished using classic statistical mechanics. Specifically, the rich, detailed information that can be collected about conversation participants in a network can be used to categorize the conversation participants. The exchanges between these categories can then be observed and analyzed to yield a meaningful description of network state.

In the Therminator model, the categories, or conversation groups, are called buckets and the exchanges between the groups are called balls. A state is defined as the number of balls in each and every bucket at any given time. The state, or bucket state concept, is further defined in Chapter III. By using the notion of a macroscopic state, the thermodynamic principles of energy, entropy, and temperature can be developed. This thesis focuses on the statistical mechanics and not the developed thermodynamics properties of the network traffic. See Reference [11] for a detailed explanation of the development of the thermodynamic properties.

As packets traverse the network, they cause balls to move between the buckets. Each packet results in the movement of a ball, and thus a new state. Each visited state is recorded for a given time period. This produces a "random walk" of the states; see Reference [12] for a detailed explanation and example. Typically, normal network traffic follows the same general walk. People tend to perform the same types of network activities each day and thus the set of exchanges is finite and follows a relatively normal pat-

tern. When a new or unusual exchange occurs, this causes one of two events. Either new states are visited that are not normally seen, or normally visited states are visited more often. Both scenarios result in an anomaly in the state walk.

For an example of this behavior, take a school of fish swimming through the ocean. Let each point in the ocean represents one possible state, with repetition, from a finite set of states. The fish tend to stay in a finite space, the school size, as they swim around. The school size represents the total observed states for a given time period. As the school swims along, all the fish follow the same path, the random walk. The path represents the change of observed states from one period to the next. When a predator approaches the school, the fish scatter, thus increasing the total observed states for that time period. This causes the anomaly in not only the walk, but the state size. When the predator leaves, the fish return to their original school size, less some unlucky fish, and continue along the random walk.

C. GUI

The Therminator GUI, pictured in Figure 1, is broken into two major parts. The upper graphic, dubbed the *Thermal Towers*, represents the average ball count in each bucket for each display period. The lower graphic, dubbed the *Thermal Canyon*, represents the frequency of bucket states for each display period. The Therminator program does have several other graphs associated with it, but those will not be explored in this thesis.

The goal of the Therminator GUI was to provide the operator a visual representation of the network traffic. Properly configured, the Therminator GUI provides insight into the "health" of the network. Just as a sonar display provides insight into the ocean environment, the Therminator GUI can provide insight into the network environment.

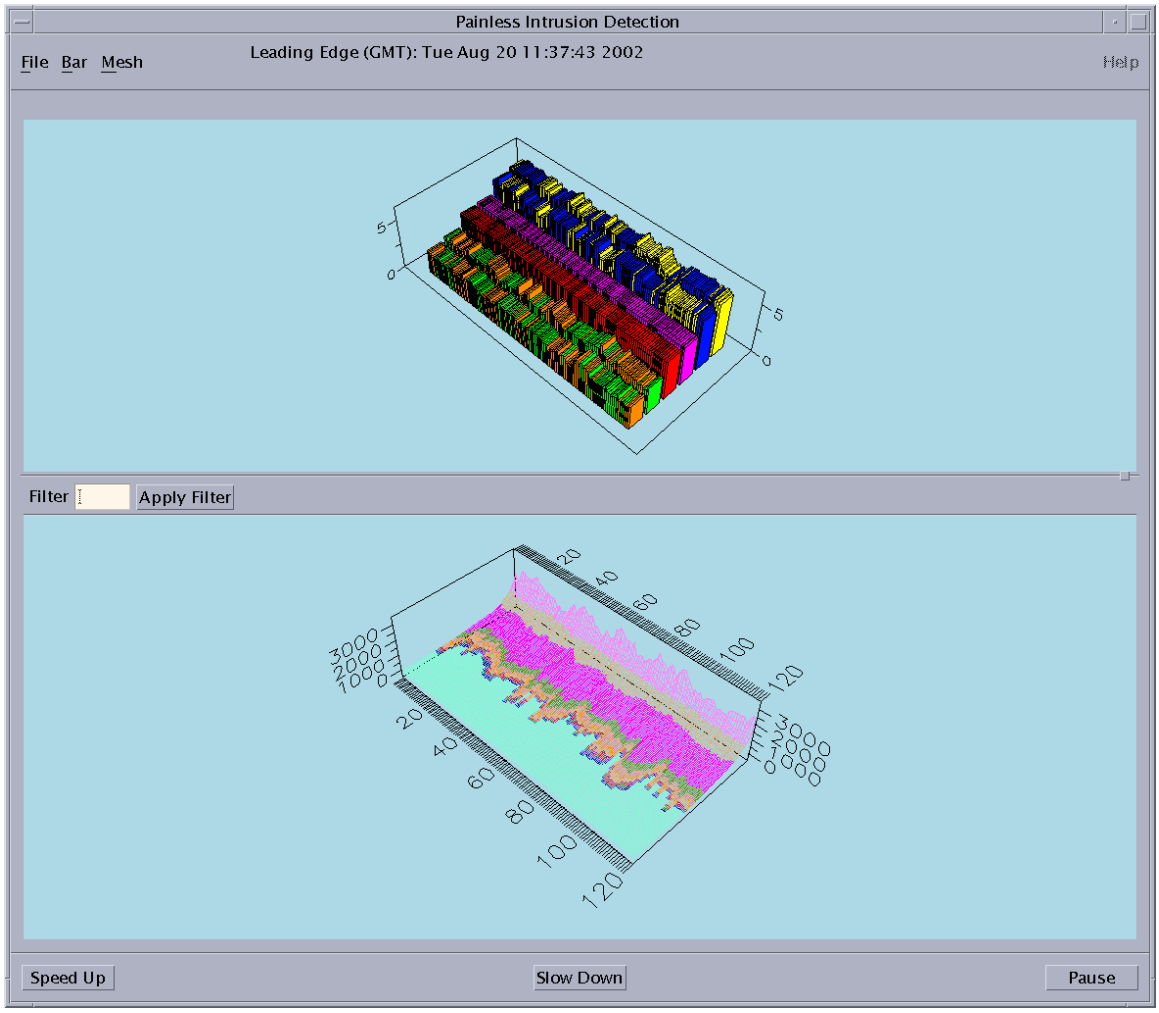


Figure 1 A screenshot of the Therminator GUI. The top half is the Thermal Towers associated with the number of balls in each bucket. The lower half is the Thermal Canyon associated with the frequency of bucket states.

1. Thermal Towers

The Thermal Towers graph lends insight into gross shifts in network traffic flow. A more detailed view of the Thermal Towers graph is shown in Figure 2. In the Thermal Towers graph, the x-axis defines time, the y-axis defines average ball count, and each colored bar along the z-axis represents a different bucket. The Thermal Towers graph allows the viewer to see changes in the average ball count from time period to time period. Generally the bars are ordered, based on average ball count, in a decreasing manner along the positive z-axis.

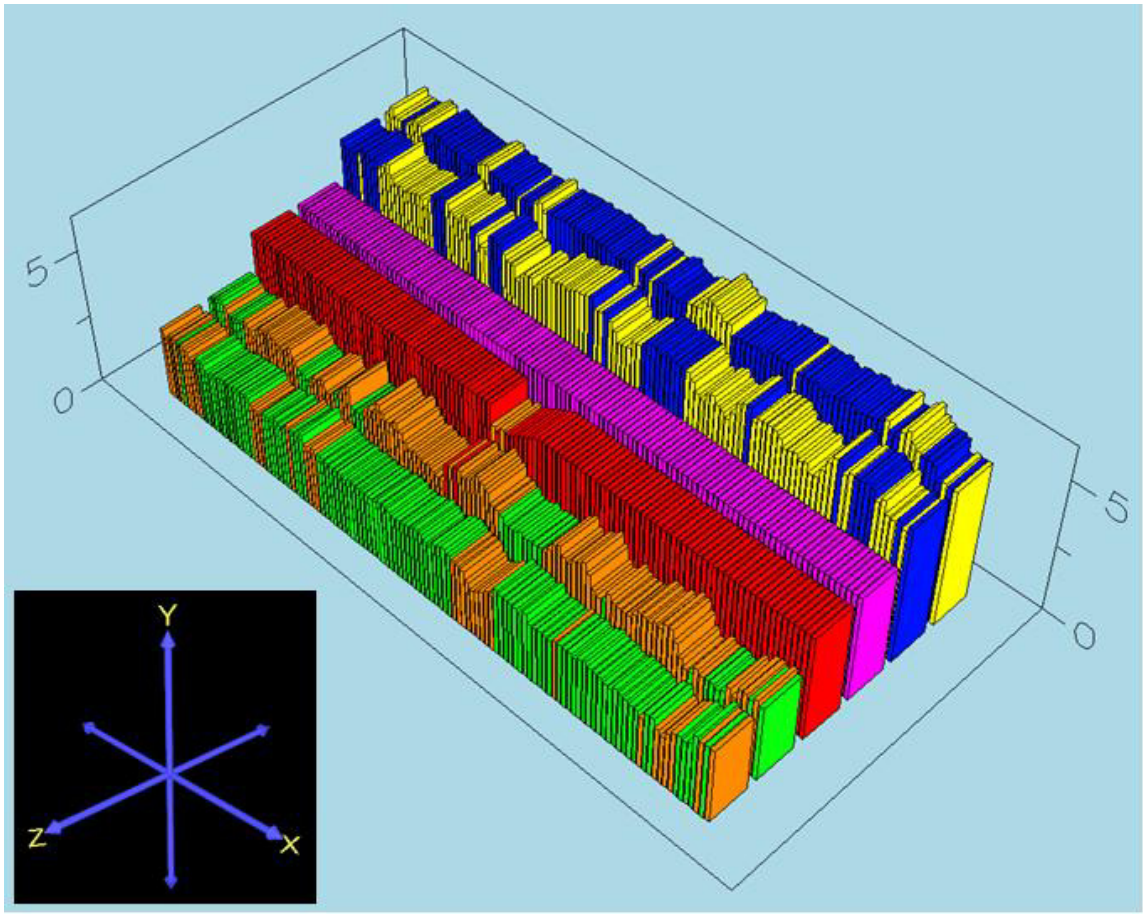


Figure 2 A screenshot example of the Thermal Towers. The x-axis represents time. The y-axis represents ball count. The z-axis represents different buckets.

2. Thermal Canyon

The Thermal Canyon graph lends insight into significant changes in the total number of unique states as well as significant changes in the total number of times a unique state is seen. A more detailed view of the Thermal Canyons graph is shown in Figure 3. In the Thermal Canyon graph, the x-axis defines time, the y-axis defines the number of times a state was seen, and each data point on the z-axis represents a unique state. The Thermal Canyon graph allows the viewer to see changes in the most probabilistic state(s) for any given time period. Color bands are used to distinguish ranges of state counts on the y-axis. These color bands make it easier to visually distinguish between significant changes in the number of times certain states are seen. All states are ordered, based on number of occurrences during the display period, in a decreasing manner along the positive z-axis.

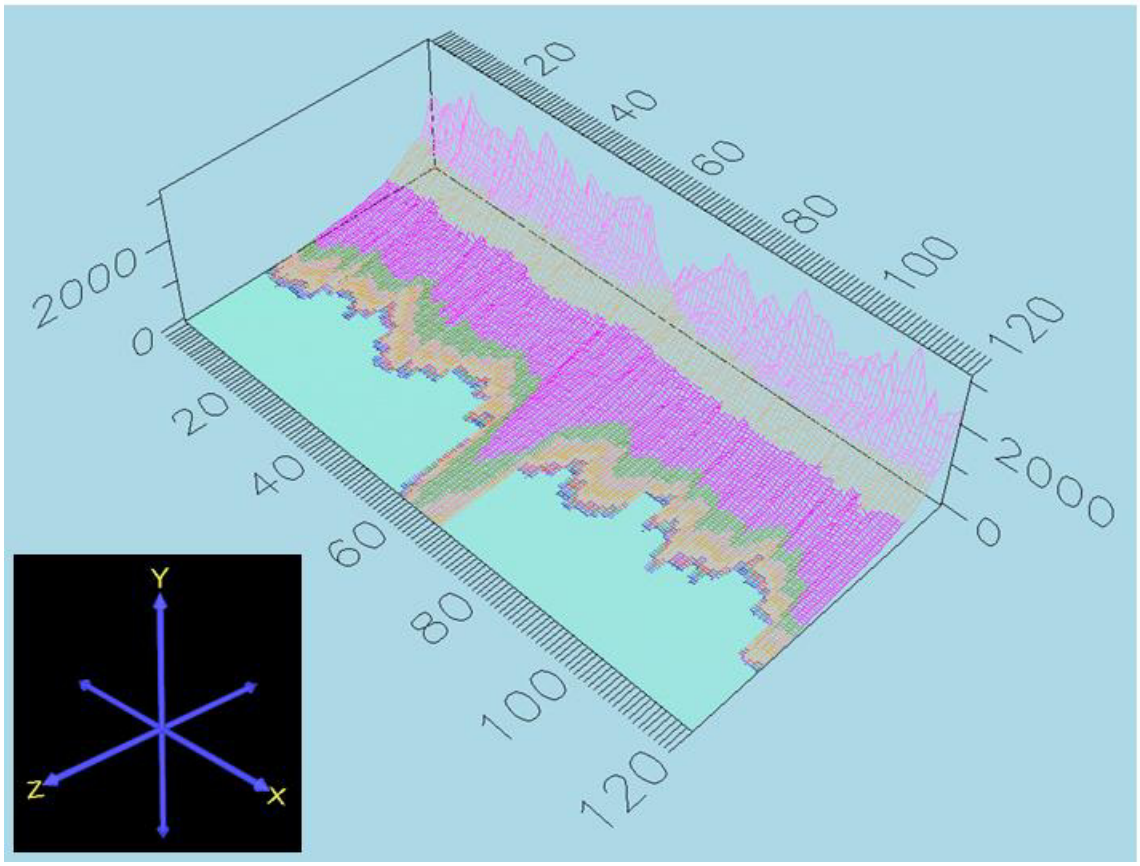


Figure 3 A screen shot example of the Thermal Canyon. The x-axis represents time. The y-axis represents bucket state count. The z-axis represents different bucket states.

D. SUMMARY

Therminator was developed as a new approach to detecting anomalous network activity. The project was developed to enable a user to get an indication of the "health" of the network by viewing two main graphs. Each graph provides insight into different characteristics of the network traffic flow. Therminator was designed to give operators a course overview of the network. The ability to detect anomalies is dependent upon the operator's aptitude for recognizing perturbations in the graphs. Humans are good at pattern recognition; so, the more pronounced the response to an anomalous event, the more likely the operator will detect it. Therminator was meant to be used in conjunction with other security tools, not replace them. In the next chapter, the concept of bucket state is defined and explored.

THIS PAGE INTENTIONALLY LEFT BLANK

III. STATES

A. CHAPTER OVERVIEW

This chapter defines the concepts of bucket state and bucket state space. The chapter then goes on to explain state counting and presents a number of equations to count the bucket states in a given bucket state space given a set of initial and boundary conditions.

B. BUCKET STATE AND BUCKET STATE SPACE

A *bucket state* is comprised of a set of *n* numbers, where *n* represents the total number of buckets in the system. Each number in the set represents the number of balls in the associated bucket at a given period in time. A representation of a bucket state is given by $\{c_1, c_2, ..., c_n\}$. The collection of allowable states forms the *bucket state space*.

C. STATE COUNTING

The bucket state space includes an initial distribution of balls among the buckets. This distribution defines the initial state, or *initial condition* for the bucket state space. Finding the total number of bucket states is a counting problem using repetition [13]. For a system comprised of *k* balls, the total number of possible bucket states, *N*, is given by the combinational mathematics equation

$$N = \binom{n+k-1}{n-1}, \tag{4.1}$$

where the initial number of balls in each bucket is the same. The special case of *k* as the total number of buckets, *n*, multiplied by the initial number of balls in each bucket, *i*, is given by

$$k = n * i. \tag{4.2}$$

An example of a bucket space is depicted in Figure 4. Ignoring the boundary conditions and applying Equation (4.1) to the system results in a total of 969 bucket states.

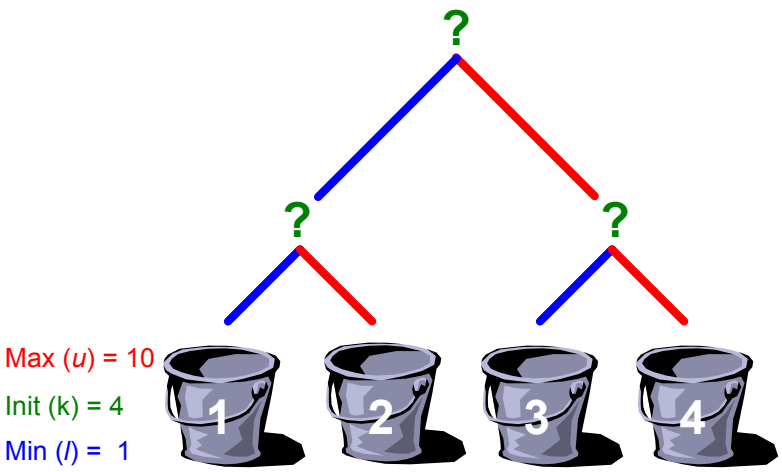


Figure 4 An example of a bucket space. In this case the bucket space is predicated by two decisions. The white numbers represent the bucket number. The initial bucket state is $\{4,4,4,4\}$.

For adherence to statistical mechanics assumptions, the allowable bucket states are bounded [1]. The *boundary conditions* are set for each individual bucket in the form of a minimum allowed ball count, l, and a maximum allowed ball count, u. The total allowable bucket states, given the same lower limit to each bucket and no upper limit is

$$N = \binom{n + (k - n \cdot l) - 1}{n - 1}. \tag{4.3}$$

Using Equation (4.2), Equation (4.3) can be restated as

$$N = \binom{n + n(i - l) - 1}{n - 1}. \tag{4.4}$$

By applying the lower limit $(l = 1)$ on the buckets, in the example above, the total number bucket states is reduced to 455.

Determining the effect of the upper limits on the bucket space is much more difficult. By subtracting out the states that violate an upper boundary condition from Equation (4.1), we have

$$N = \binom{n + k - 1}{n - 1} - \sum_{j=1}^{\left\lfloor k/(u+1) \right\rfloor} (-1)^{j+1} \binom{n}{j} \binom{n + (k - j(u+1)) - 1}{n - 1}. \tag{4.5}$$

This assumes that each bucket has the same upper limit. Applying Equation (4.5) to the example above, with an upper limit (u) of ten, reduces the total number of bucket states to 745.

Combining the upper and lower limits leads to

$$N = \binom{n + n(i-l) - 1}{n-1} - \sum_{j=1}^{\lfloor n(i-l)/(u-l+1) \rfloor} (-1)^{j+1} \binom{n}{j} \binom{n + (n*i) - j(u+1) - \sum_{j+1}^{n} l - 1}{n-1}. \quad (4.6)$$

Applying the boundary conditions defined in the example above, the total number of buckets states is further reduced to 415.

In order to determine the total allowable bucket states, given unique boundary conditions and initial conditions for each bucket results in

$$N = \binom{n + \sum_{m=1}^{n} (i_m - l_m) - 1}{n-1} - \sum_{j=1}^{z} (-1)^{j+1} \binom{n}{j} \binom{n + \sum_{m=1}^{n} i_m - \sum_{m=j+1}^{n} l_m - \sum_{m=1}^{j} (u_m + 1) - 1}{n-1} \quad (4.7)$$

and

$$z = \left\lfloor \frac{\sum_{m-1}^{n} (i_m - l_m)}{\sum_{p-1}^{n} (u_p - l_p + 1)} \right\rfloor. \quad (4.8)$$

In these equations, i_j and l_j represent the initial number of balls and the minimum number of balls for the j-th bucket respectively. The maximum allowed balls in a given bucket is represented by u_j.

D. SUMMARY

Bucket states are the basic building block from which the Thermintor GUIs are derived. Understanding the expected bucket state space and the total number of possible bucket states, for a given configuration, is important for proper Therminator configuration and interpretation. The role of total bucket states will be developed in the Chapter V. This chapter also defined equations for determining the total possible bucket states under a variety of boundary conditions. In the next chapter, the details of the test network are described.

THIS PAGE INTENTIONALLY LEFT BLANK

IV. TEST NETWORK

A. CHAPTER OVERVIEW

In this chapter, the design and setup of the test network used to perform all of the experiments is described. This includes the equipment used, network topology, and traffic composition.

B. TEST NETWORK DESCRIPTION

All experiments were run from traffic generated on the test network depicted in Figure 5. This test network was used so that the traffic characteristics could be closely controlled and reproduced.

The network was intended to simulate a medium sized network, dubbed the trusted network, connected to the Internet and untrusted networks. For traffic generation, a Spirent SMB-6000B chassis [14] with five LAN-3302A TeraMetrics modules [15] were used. The LAN-3302A Ethernet modules are capable of generating millions of simultaneous TCP/IP connections at full line rate. In conjunction with the Avalanche SMB software [16], the system can simulate millions of clients and servers on multiple subnets, thus generating realistic Internet conditions and load. The software provides for the simulation of numerous layer 2-7 protocols, including HTTP, which was mainly used for this research. The behavior of the servers and clients is also fully configurable with the Avalanche SMB software. The only exception is the distribution of clients between the trusted and untrusted networks. The program uses a round robin technique to assign clients to the different subnets. This lack of functionality can be overcome by configuring more subnets for untrusted clients than trusted clients in the desired distribution ratio. For the purposes of these experiments, this was not an issue.

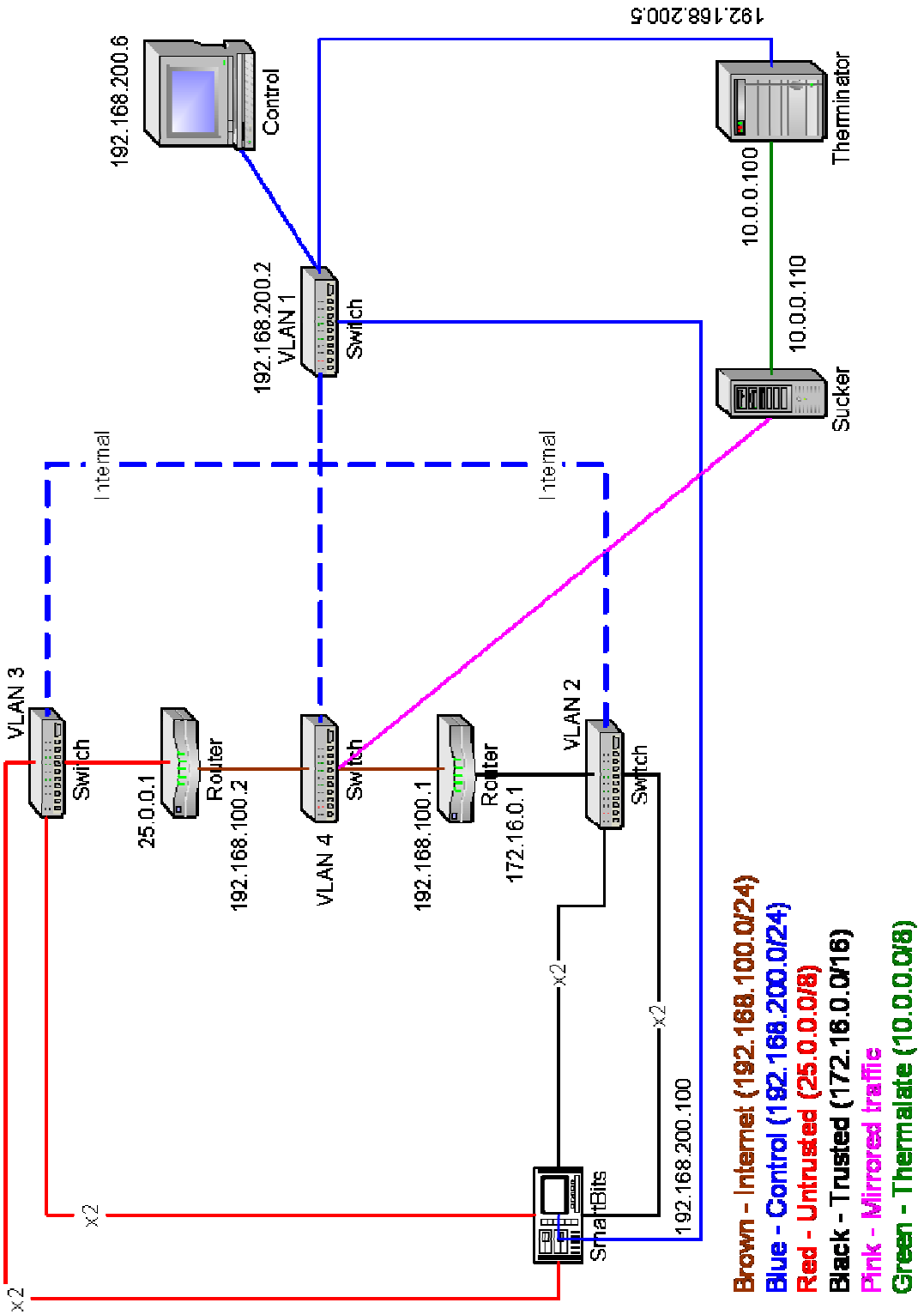


Figure 5 A schematic of the test network. The network is comprised of a trusted and untrusted side, divided by two routers. The core of the traffic generation is a Spirent Smartbits traffic generator.

The test network was divided in two by a pair of Cisco 2600 routers, one acting as the trusted network's gateway or edge device and the other as the first hop in the Internet. The traffic passing through the routers is captured and sent to the Therminator IDS. Two routers were used, as opposed to one, to provide a more realistic simulation of a network's connection to the Internet, since most networks consist of more than one router. By using two routers, network packets addressed to non-existent machines will be passed into the network. Typically it is the last hop router that generates the *destination host unreachable ICMP* error, thus using two routers at the gateway ensures that invalid traffic passes through the capture point. A Cisco Catalyst 4000 switch, partitioned into four VLANs, was used for connecting multiple inputs to each router interface. The uses of each VLAN are outlined in Table 1.

VLAN	Description	Network	Use
VLAN1	Control network	192.168.200.0/24	Used for administrative control of all devices in the test network
VLAN2	Trusted Network	172.16.0.0/16	Simulated medium sized network connected to the Internet (single autonomous system)
VLAN3	Untrusted Network	10.0.0.0/8	Simulated any number of untrusted networks connected to the Internet (multiple autonomous systems)
VLAN4	Internet	192.168.100.0/24	Simulated Internet (connecting autonomous systems) and provides capability for mirroring network traffic entering and exiting the trusted network

Table 1 A table containing the test network VLAN descriptions including network address and purpose

The Avalanche SMB software allows each LAN-3302A module to be configured as either *Web Avalanche* (clients) or *Web Reflector* (servers). The trusted network was broken into four subnets – two for servers and two for clients. The untrusted network(s) also consisted of four subnets broken down in the same fashion. The configuration of each of the five LAN-3302A modules and associated subnets can be seen in Table 2.

Module #	Sofware	Network	Subnet	Clients/Servers
(0,0,0)	Web Reflector	Untrusted	25.0.10.0/24	198 Servers
(0,0,1)			25.0.20.0/24	198 Servers
(0,1,0)	Web Avalanche		25.1.0.0/16	199 Clients
(0,1,1)			25.2.0.0/16	199 Clients
(0,2,0)	Web Reflector	Trusted	172.16.10.0/24	5 Servers
(0,2,1)			172.16.20.0/24	5 Servers
(0,5,0)	SmartWindow			1 Server
(0,3,0)	Web Avalanche		172.16.100.0/24	50 Clients
(0,3,1)			172.16.200.0/24	50 Clients

Table 2 A table containing the configuration information for the five LAN-3302A modules used in the Spirent Smatbits traffic generator.

The module numbers are based on location in the SMB-6000B chassis. The first number is the chassis number, 0 in this case, since chassis can be daisy chained together. The second number is the slot in the chassis, starting with 0 in the upper left corner and counting across, row by row. The third number is the port number on a given module. Each module has two 10/100 Base-TX ports, labeled from left to right starting at 0.

SmartWindow [17] is a virtual front panel for the LAN-3302A modules that allows for custom packet generation and insertion. Module (0,5,0) was used for generation and injection of anomalous packets into the trusted network.

C. SUMMARY

All experiments were performed using traffic generated by a test network. The test network allowed for close control over the type of traffic generated. It also provided a means for reproducing the same type of traffic for multiple trials, in order to compare the results of different configuration changes. The next chapter contains experiments that explore the SLIDELENGTH and WINDOWLENGTH configuration parameters.

V. CONFIGURATION – SLIDELENGTH AND WINDOWLENGTH

A. CHAPTER OVERVIEW

In this chapter, the results of numerous experiments that explore the SLIDELENGTH (SL) and WINDOWLENGTH (WL) configuration parameters will be discussed. The concept of the smoothing ratio will be defined and explored. The experiments include determining the optimum traffic rate for a given set of configuration parameters and the effects of varying the SL, WL, and smoothing ratio. Another set of experiments explored configuration for a traffic rate outside the optimal range. The last set of experiments looked at the effect of unequal traffic distribution among buckets.

B. INTRODUCTION

Two key configuration parameters are the SL and the WL. The SL is defined as the time, in seconds, used for one display period. The WL is the total period of time, in seconds, that data is averaged over. For example, a SL of two means that data is collected in two second intervals with each display increment representing two seconds. With a WL of ten, those two seconds of data would be averaged over ten seconds, or five SLs. An example can be seen in Figure 6.

In this example, the SL is the difference between time marks. The WL is the difference between the start and finish of a box. Using this example, the data displayed at time t_1 includes the data from time t_{-4} through time t_0.

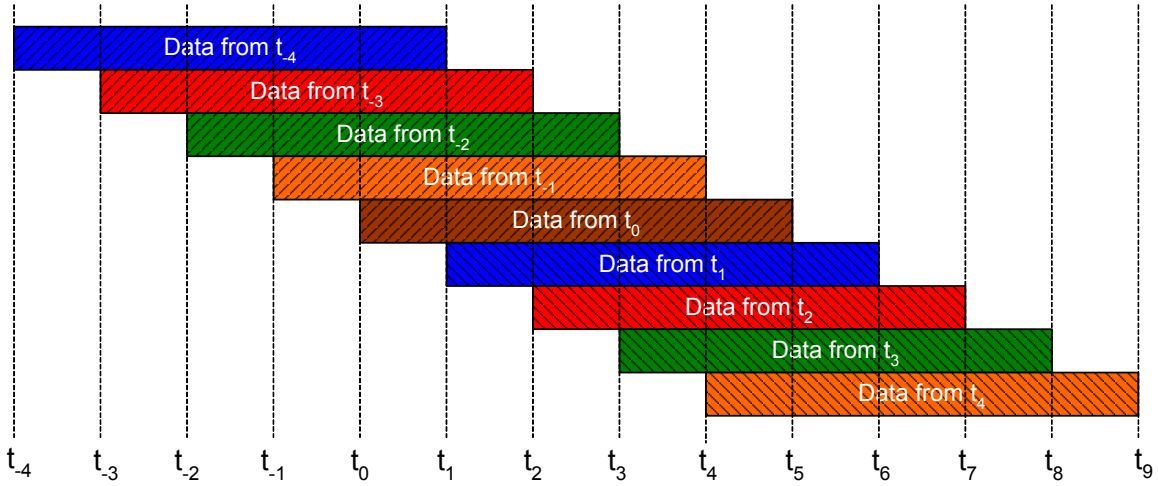


Figure 6 A graphic showing the relationship between the SL and WL. The SL is the distance between any two time periods. The WL is the distance spanned by any bar.

C. SMOOTHING RATIO

The number of display periods that a given set of data will be averaged over is called the *smoothing ratio* (SR). The value of SR is calculated as

$$SR = \frac{WINDOWLENGTH}{SLIDELENGTH}. \tag{6.1}$$

The SR is analogous to a simple filter. The purpose of a filter is to reduce unwanted oscillations in a system. In this case, the SR reduces the noise generated by a single event by averaging it over a set time period. The SR must be set properly for a graph to reveal any useful information. A large ratio will tend to hide anomalies (false negatives), where a small ratio will result in normal traffic appearing anomalous (false positives).

An example of a small smoothing ratio, SR, equal to one, can be seen in Figure 7 (a) and (b). These graphs are very rough, with most activity looking anomalous. An example of a high smoothing ratio, SR equal to fifty, can be seen in Figure 7 (c) and (d). In these graphs, all of the useful information has been smoothed out. It would be nearly impossible to see any anomalous activity in these graphs. Last, the set of graphs in Figure 7 (e) and (f), the SR is set at ten. This has the effect of filtering out the unwanted noise, while maintaining the desired system information.

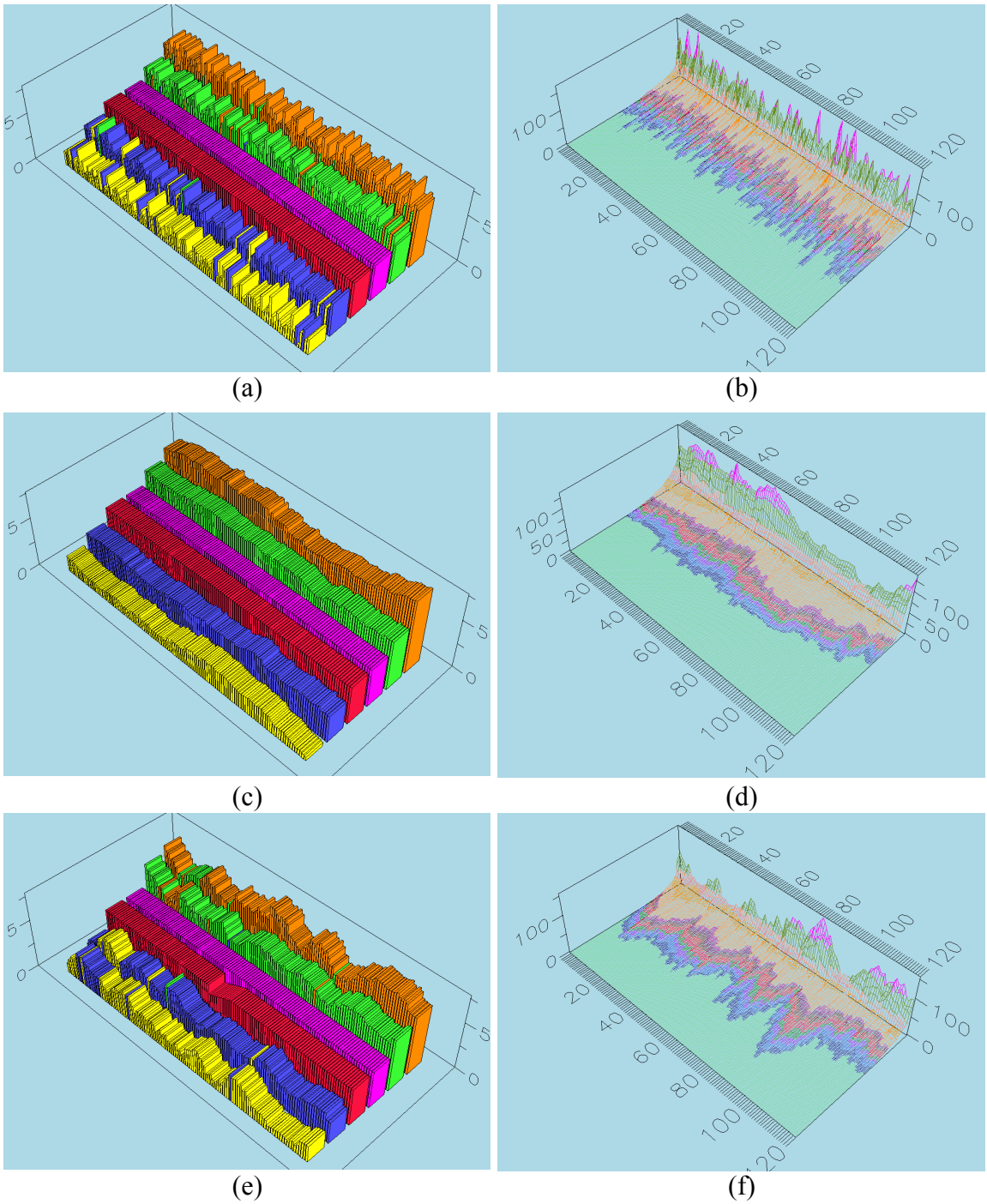


Figure 7 Screen shots of Therminator graphs depicting the effects of various smoothing ratios – (a) Thermal Towers with SR = 1, (b) Thermal Canyon with SR = 1, (c) Thermal Towers with SR = 50, (d) Thermal Canyon with SR = 50, (e) Thermal Towers with SR = 10, (f) Thermal Canyon with SR = 10

D. SLIDELENGTH AND WINDOWLENGTH

The SL controls how long a display period is, thus controlling how many new packets are analyzed per display period (*new packet count*). The total number of new packets analyzed per display period (T_{np}) is given by

$$T_{np} = R_p\left(SLIDELENGTH\right), \tag{6.2}$$

where R_p is the packet rate of the network.

The WL controls how long data is averaged over, or the total number of packets analyzed per display period (*absolute packet count*). The total number of packets analyzed per display period (T_p) is given by

$$\begin{aligned} T_p &= R_p\left(SLIDELENGTH * SR\right) \\ &= R_p\left[SLIDELENGTH\left(\frac{WINDOWLENGTH}{SLIDELENGTH}\right)\right] \\ &= R_p\left(WINDOWLENGTH\right). \end{aligned} \tag{6.3}$$

The SL and WL must be set properly in order to obtain a sufficient amount of traffic per display period to give any meaningful results. In order to determine a guideline for proper setting of these parameters, a series of experiments were run. Both the SL and WL can be found in the <name>.config file for the Therminator executable PID5. Appendix A contains an example of each of the Therminator configuration files.

E. RATE EXPERIMENT

The first experiment was run using the parameters shown in Table 3.

SLIDELENGTH	WINDOWLENGTH	Smoothing Ratio	BucketSpaceInit
1 second	10 seconds	10	0/10 : 4

Table 3 A table containing the configuration parameters for the rate experiment

The traffic rate was varied in order to determine how the system reacted to the various traffic rates. The traffic was composed of simple HTTP transactions, each consisting of an eight packet exchange, similar to that seen in Figure 8.

	Time	Source	Destination	Protocol	Info
9	0.066028	172.16.200.51	25.0.10.4	TCP	14312 > http [SYN] Seq=15
0	0.066785	25.0.10.4	172.16.200.51	TCP	http > 14312 [SYN, ACK] S
1	0.066788	172.16.200.51	25.0.10.4	TCP	14312 > http [ACK] Seq=15
2	0.066909	172.16.200.51	25.0.10.4	HTTP	GET / HTTP/1.1
3	0.068003	25.0.10.4	172.16.200.51	HTTP	HTTP/1.0 200 OK
4	0.068007	25.0.10.4	172.16.200.51	TCP	http > 14312 [FIN, ACK] S
5	0.068028	172.16.200.51	25.0.10.4	TCP	14312 > http [FIN, ACK] S
6	0.069106	25.0.10.4	172.16.200.51	TCP	http > 14312 [ACK] Seq=15

Figure 8 A screen shot of an Ethereal capture showing a typical HTTP packet exchange generated by the test network. The exchange consists of eight packets.

By setting the load parameter in the Avalanche SMB software to users per second, the traffic rate could be controlled. Using a constant of 8 packets per user per second, any traffic rate could be supplied by adjusting the number of users per second. This produced a constant packet rate with less than three percent standard deviation. Each user that is created will perform one transaction, similar to that shown in Figure 8. Since each transaction lasts approximately 3 ms, each user exits for only 3 ms.

A simple bucket space definition was used which is depicted in Figure 9. The system consisted of six buckets.

Figure 9 A screen shot of the Therminator bucket space definition used in all experiments. The bucket space consists of six buckets. The "!" represents a negation. Services represent port numbers less than 1024.

The blue bucket represents the category of web servers within the trusted network using services ports – ports less than 1024. The red bucket represents non-services ports, ephemeral, on web servers within the trusted network. The Magenta bucket represents trusted clients service ports and the yellow bucket represents ephemeral ports on trusted clients. The dark orange bucket represents service ports on hosts in the untrusted network. Last, the green bucket represents ephemeral ports on untrusted hosts.

For the purpose of this experiment, the test traffic only causes balls to move between the green and blue buckets as well as the yellow and dark orange buckets. The traffic rates between these two bucket pairs should be equally distributed as the Avalanche SMB software evenly populates the trusted and untrusted subnets using a round robin algorithm.

Given that there are two sets of buckets with balls moving only within each set. The total number of possible states can be determined by using the BucketSpaceInit parameters in Table 3 with Equations (4.7) and (4.8). In this case, the total number of buckets is 2, one bucket pair. The results should be squared to account for both bucket pairs. This results in 81 total possible states. The Thermal Canyon can display 100 unique states before the states begin to run off the end of the display.

Each experiment shows a single packet anomaly that is orthogonal to the normal traffic flow. This causes a ball to move from the red to the dark orange bucket. By adding a ball to the orange bucket, the total possible states are increased to 90. Given that the data is carried for 10 display periods, there are actually 171 possible states that can be seen during a WL of time after the anomaly.

The comments and conclusions drawn from these experiments are valid for the given bucket space definition in Figure 9 and the configuration parameters in Table 3. The results of changing these parameters are explored in Chapter VI.

For this experiment, the traffic rate was varied from 250 packets per second (pps) to 10 kpps. With a SL of 1 second, the new packet count per display period is equal to the traffic rate. The absolute packet count per display period is equal to 10 times the traffic rate; see Equation (6.3).

A traffic rate of 250 pps produces a graph that is difficult to read due to the low traffic rate. Each exchange weighs heavily upon the graph causing a steep ramping effect as seen Figure 10(a). The ramping effect is caused by the SR. With the low traffic rate, there are a small number of state changes and each state change is averaged over an SR number of display periods. In this case, the SR actually creates “noise”. The Low traffic

rate also causes only a small percentage of the total states to be visited as seen in Figure 10(b).

Doubling the traffic rate improves the graph dramatically, but it is still very rough, as shown in Figure 10(c). This rough nature is caused by the ramping effect as seen in Figure 10(a), but with the higher traffic rate, only a partial ramp is allowed to develop. The added noise in the system is thus reduced from that of the previous traffic rate. The traffic rate is not high enough to see a large percentage of the possible states visited in the Thermal Canyon, thus Figure 10(d) does not produce much useful information.

At 1 kpps, the Thermal Towers begin to show useful information about the conversation groups. The graph is still rough, as shown in Figure 10(e), but has smoothed out considerably. The graph is still hard to interpret due to the excess noise present in the system. As more states are visited, the Thermal Canyon, Figure 10(f), begins to reveal useful information.

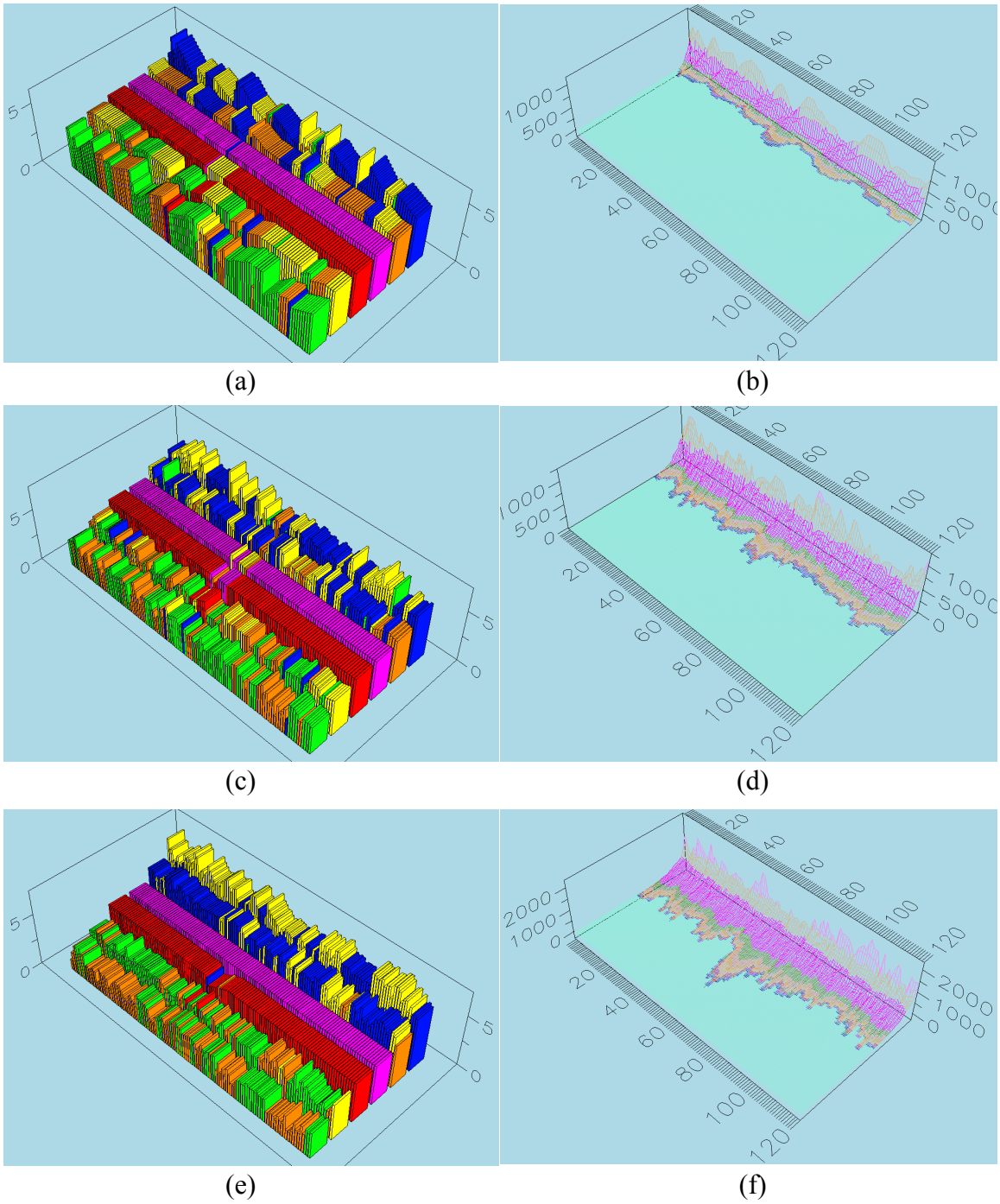


Figure 10 Screen shots of the rate experiment results – (a) Thermal Towers at 250 pps, (b) Thermal Canyon at 250 pps, (c) Thermal Towers at 500 pps, (d) Thermal Canyon at 500 pps, (e) Thermal Towers at 1 kpps, (f) Thermal Canyon at 1k pp.

As the traffic rate reaches 2 kpps, the graphs became smoother and easier to interpret. The conversation groups are easily distinguished in the Thermal Towers shown in Figure 11(a). The number of states visited has reached a level that small perturbations in the normal traffic flow are readily apparent as shown in Figure 11(b). A traffic rate of 2 kpps rate marks the low end of the desirable traffic rate for proper performance.

With a traffic rate of 3 kpps, the conversation groups in the Thermal Towers are very easy to distinguish as the graph has smoothed out significantly. This is depicted in Figure 11(c). As a high percentage of the possible states are now being visited, the Thermal Canyon graph yields rich information, as shown in Figure 11(d). The single packet anomaly almost doubles the number of visited states. This produces a perturbation in the graph that is easy to detect. A traffic rate of 3 kpps marks the high end of the desirable traffic rate for proper performance.

As the traffic rate was increased past 3 kpps, the system became overly smooth and the conversation groups disappear, as shown in Figure 11(e). At these traffic rates, nearly all of the states are visited each display period, and then averaged over the entire WL. This results in very little change in the average ball count of a given bucket from display period to display period. The Thermal Canyon, shown in Figure 11(f), is extremely smooth as the same number of unique states is visited in each display period. The single packet anomaly is still easy to distinguish, but more subtle network fluctuations are lost. As the traffic continues to increase, the system approaches complete saturation. At this point, every possible state is visited each display period. At these high traffic rates, the system yields no useful information.

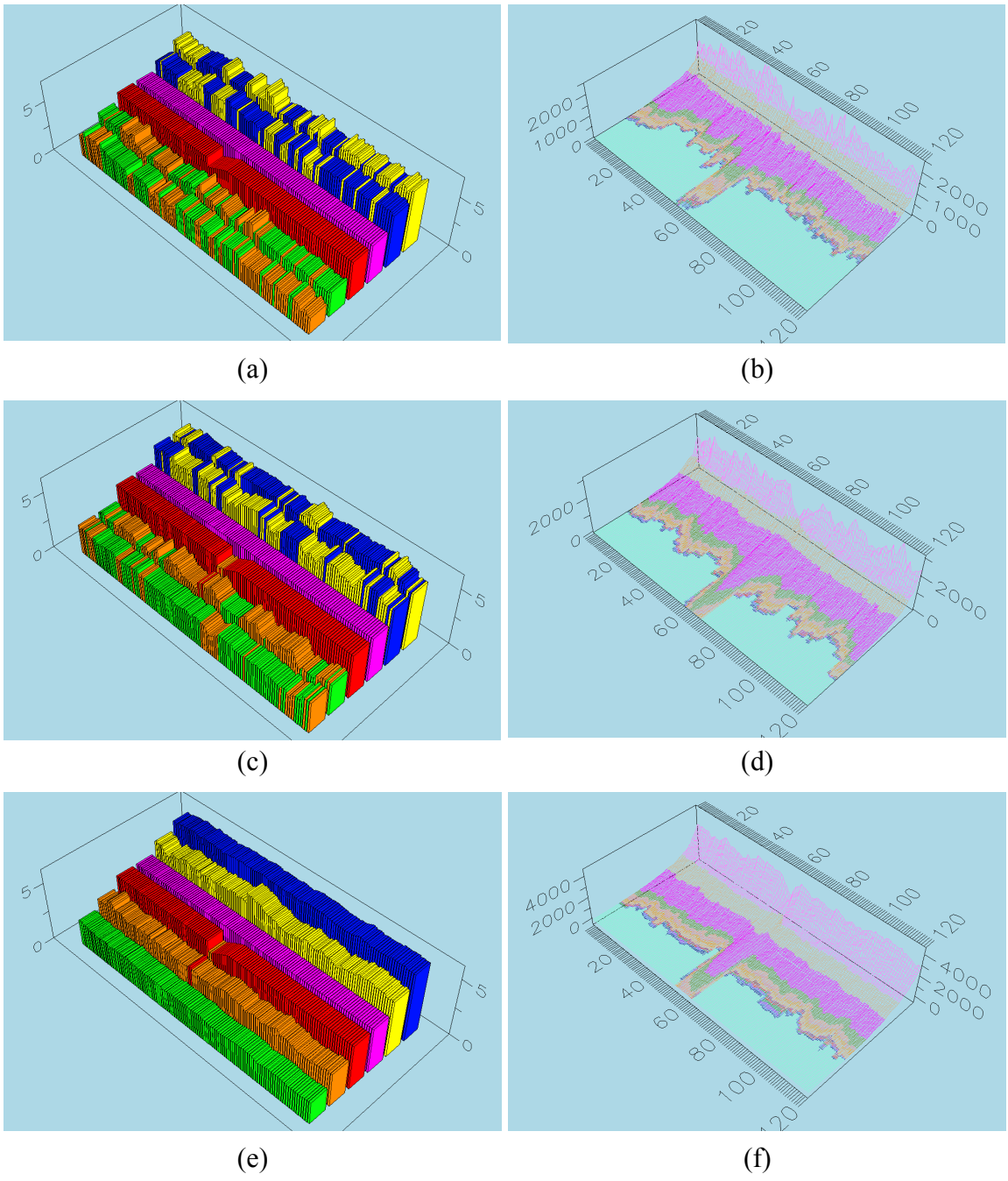


Figure 11 Screen shots of the rate experiment results – (a) Thermal Towers at 2 kpps, (b) Thermal Canyon at 2 kpps, (c) Thermal Towers at 3 kpps, (d) Thermal Canyon at 3 kpps, (e) Thermal Towers at 5 kpps, (f) Thermal Canyon at 5 kpps

1. Conclusions

From this first experiment, it is apparent that a traffic rate of 2 – 3 kpps is desirable for proper operation, given the configuration parameters in Table 3. In order to apply this as a general configuration rule, there are a few questions that must first be explored.

1. Is the desired traffic rate based on the new packet count for each display period (SL) or on the absolute packet count (WL)?
2. Is this rate valid for different SR values?
3. Do the SL and WL parameters scale appropriately to produce an effective graph at lower/higher traffic rates?

F. RATIO EXPERIMENT

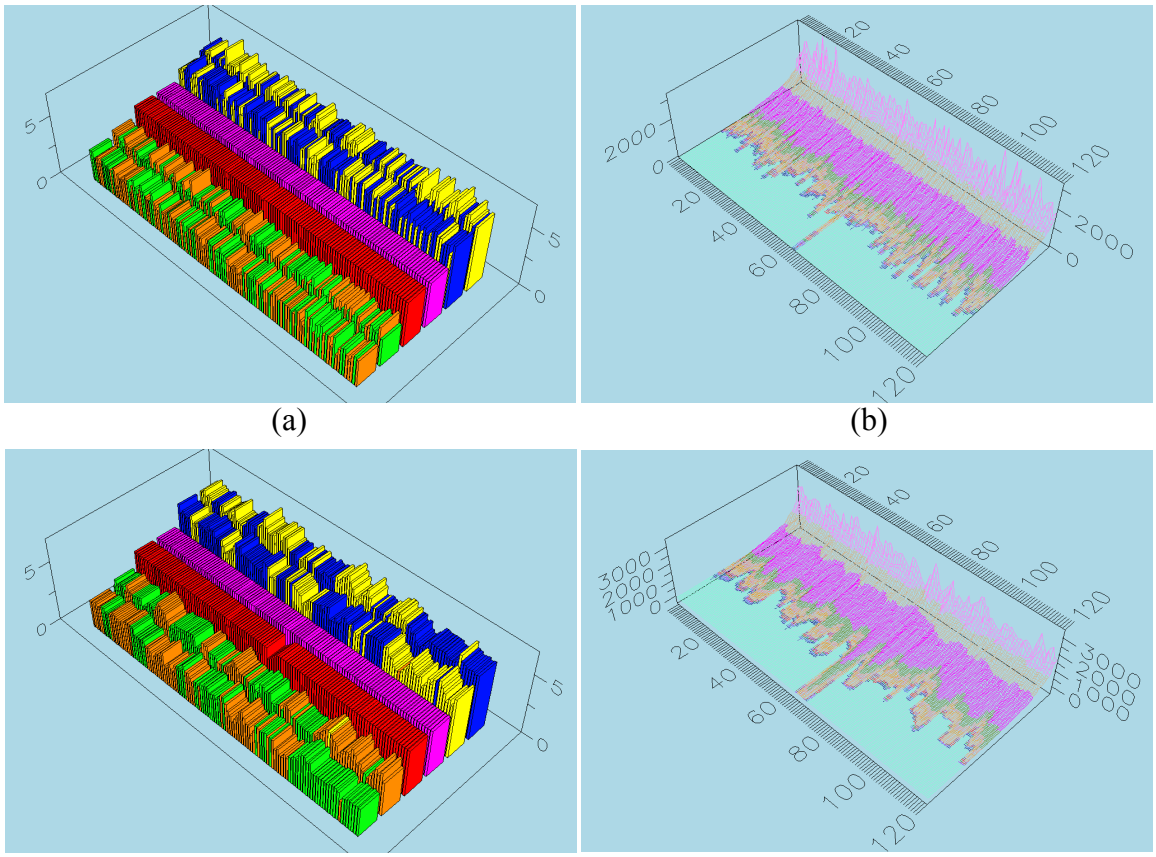
1. Constant WINDOWLENGTH

The next experiment was developed to answer question one above. A constant traffic rate of approximately 2.4 kpps was used based on the results of the previous experiment. This rate was accomplished with a 300 user-per-second load on the Avalanche SMB software. The experiment consisted of setting the WL, using Equation (6.3), to achieve an absolute packet count of 24 thousand packets per window. The experiment was conducted four times, each with a different SR. The SL was set, using Equation (6.1), to achieve an SR of 1, 5, 10, and then 20. The setting and corresponding figures for each of the four experiment runs are shown in Table 4.

WINDOWLENGTH	SLIDELENGTH	SR	Figures
10	10	1	Figure 12 a & b
10	2	5	Figure 12 c & d
10	1	10	Figure 13 a & b
10	0.5	20	Figure 13 c & d

Table 4 A table containing the configuration parameters for the constant WL rate experiment and the associated figures depicting the results.

Comparing the eight figures listed in Table 4, it can be seen that they are dramatically different. If the response of the system was based on the absolute packet count, then each of these figures would look similar. In actuality, these figures are similar to those in the section V.E rate experiment. The figures with a low SR exhibit a lot of noise, as did the figures with low traffic rates. The figures with the higher SRs approach a saturation condition as did the figures with high traffic rates.



(a) (b)

(c) (d)

Figure 12 Screen shots of the constant WL ratio experiment results – (a) Thermal Towers with SR = 1, (b) Thermal Canyon with SR = 1, (c) Thermal Towers with SR = 5, (d) Thermal Canyon with SR = 20

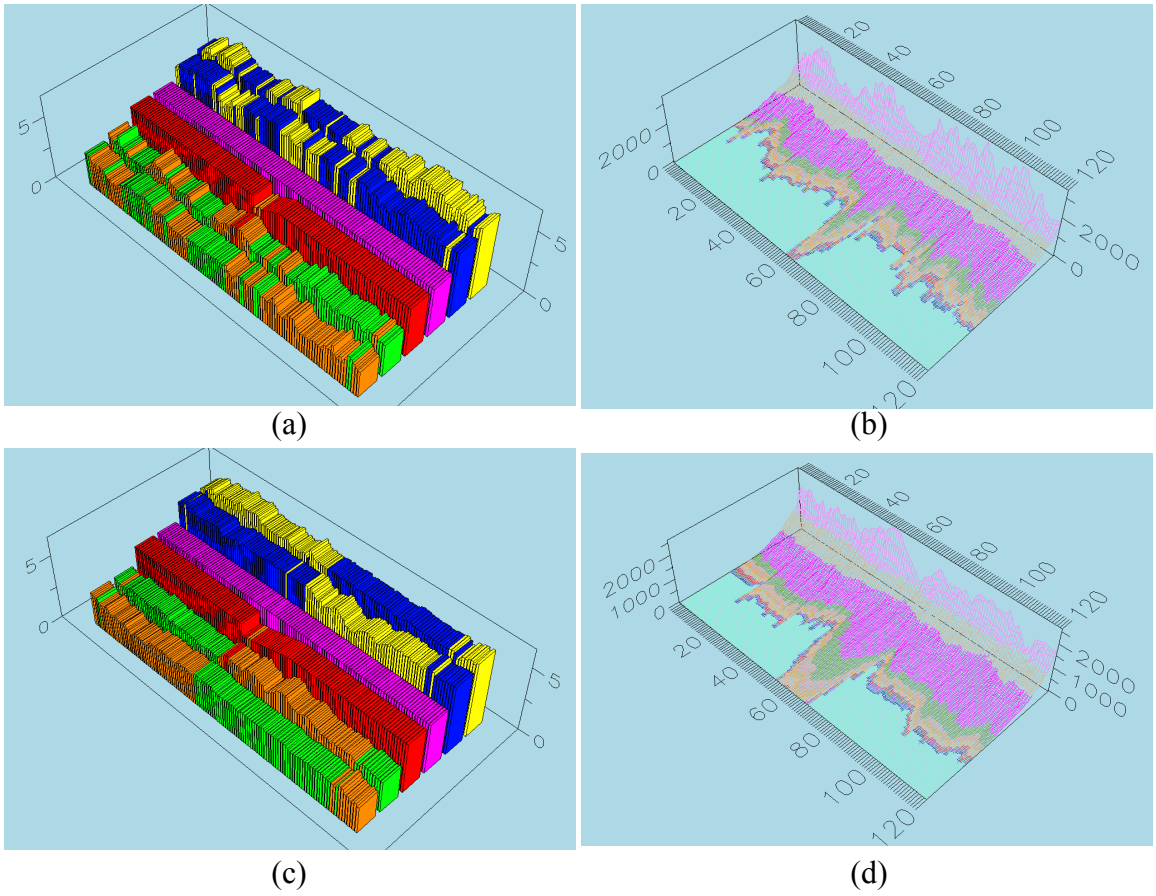


Figure 13 Screen shots of the constant WL ratio experiment results – (a) Thermal Towers with SR = 10, (b) Thermal Canyon with SR = 10, (c) Thermal Towers with SR = 20, (d) Thermal Canyon with SR = 2.

a. Conclusions

It can be concluded that optimal system operation is based on the new packet count as opposed to absolute packet count. Therefore, the SL must be set first to achieve the correct new packet count, and then the WL can be set to achieve the desired SR.

2. Constant SLIDELENGTH

Given that the SL must be fixed, the next experiment explored question two from section V.E.1, of whether the optimal new packet count of 2 – 3 thousand packets per display period is valid for different SRs. The experiment consisted of setting the SL to achieve a new packet count of 2.4 thousand packets per slide. The experiment was conducted four times, each with a different SR. The WL was set to achieve an SR of 1, 5,

10, and then 20. The setting and corresponding figures for each of the four runs of the experiments are shown in Table 5.

WINDOWLENGTH	SLIDELENGTH	SR	Figures
1	1	1	Figure 14 a & b
5	1	5	Figure 14 c & d
10	1	10	Figure 15 a & b
20	1	20	Figure 15 c & d

Table 5 A table containing the configuration parameters for the constant SL rate experiment and the associated figures depicting the results.

Comparing the eight figures listed in Table 5, it can be seen that they are dramatically different. If the response of the system was independent of the SR, then each of these figures would look similar. The experiment results are as expected, given that the SR acts as a filter. The figures with a low SR are very noisy, where the figures with the higher SR are much smoother, producing a usable result.

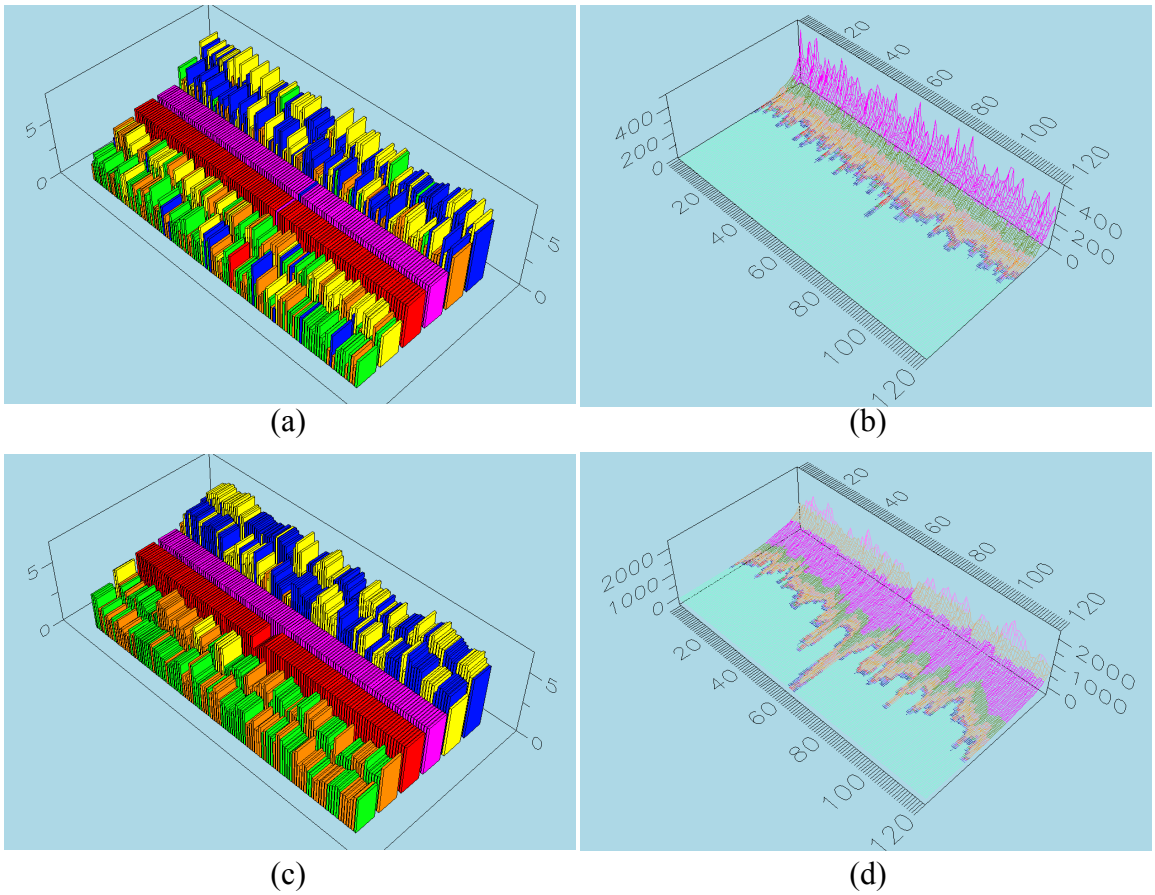


Figure 14 Screen shots of the constant SL ratio experiment results– (a) Thermal Towers with SR = 1, (b) Thermal Canyon with SR = 1, (c) Thermal Towers with SR = 5, (d) Thermal Canyon with SR = 5

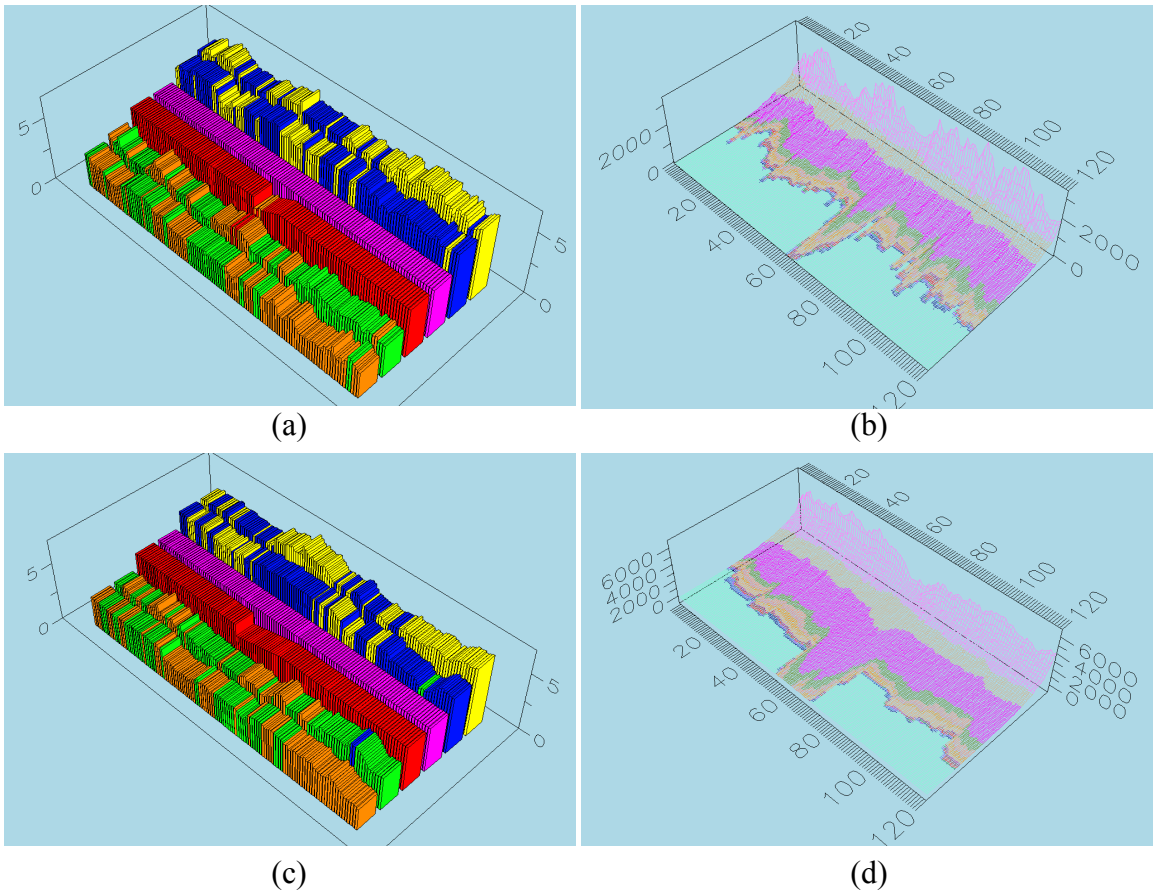


Figure 15 Screen shots of the constant SL ratio experiment results– (a) Thermal Towers with SR = 10, (b) Thermal Canyon with SR = 10, (c) Thermal Towers with SR = 20, (d) Thermal Canyon with SR = 20

a. Conclusions

It can be concluded that optimal system operation is dependent on the proper new packet count as well as the proper SR. Once the SL is properly set, then the WL must be set to achieve the optimal SR. For network traffic rates between 2 and 3 kpps, an SR of approximately 10 results in an exceptional response from the Thermal Towers and Thermal Canyon graphs.

G. LOW DATA RATE EXPERIMENT

Given that not all networks have traffic levels between 2 and 3 kpps, this section explores configuration for lower data rates – specifically how the SL and WL parameters scale with lower traffic levels. To accomplish this, a series of experiments were developed. The first experiment checked the scalability of the SL. The second experiment

checked the scalability of the SR as well as the WL. All experiments were performed using a traffic rate of 500 pps.

1. SLIDELENGTH Scaling Experiment

The SL Scaling experiment was designed to explore the scalability of the SL parameter. If a SL of 1 worked well for traffic rates of 2 to 3 kpps, then is there a scaling relation for other traffic rates? If a scaling relationship exists, then it may not be linear. In Figure 16, curve 'a' shows an example of polynomial scaling and curve 'b' shows an example of linear scaling.

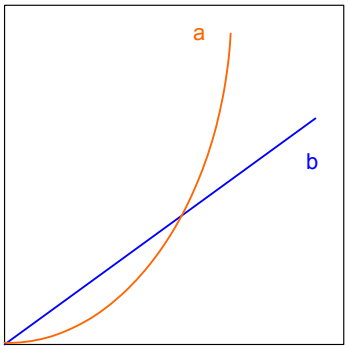


Figure 16 A graphic depicting an example of polynomial (a) and linear (b) scaling.

For this experiment, the SL was varied from 3 to 5 and the WL was changed to ensure a SR of 10. The values of SL, WL, SR, and the corresponding figures for each run of the experiment are outlined in Table 6.

WINDOWLENGTH	SLIDELENGTH	SR	Figures
30	3	10	Figure 17 a & b
40	4	10	Figure 17 c & d
50	5	10	Figure 17 e & f

Table 6 A table containing the configuration parameters of the low data SL scaling experiment and the associated figures depicting the results.

If there existed a linear relationship between SL and traffic rate, then an SL value of 5 would produce the same results for a traffic rate of 500 pps as an SL of 1 did for a traffic rate of 2.5 kpps. It can be seen in Figure 17(e) and (f) that this is not the case. When compared to Figure 13(a) and (b), it can be seen that they are dramatically different. The graphs in Figure 17(e) and (f) have been overly smoothed, thus resulting in lost information. This would indicate that there is not a linear relationship. One reason is that the transaction time does not change with the traffic rate. The transaction time is a con-

stant 3 ms; thus, for a longer SL, a larger percentage of transaction are completed as compared to a shorter SL.

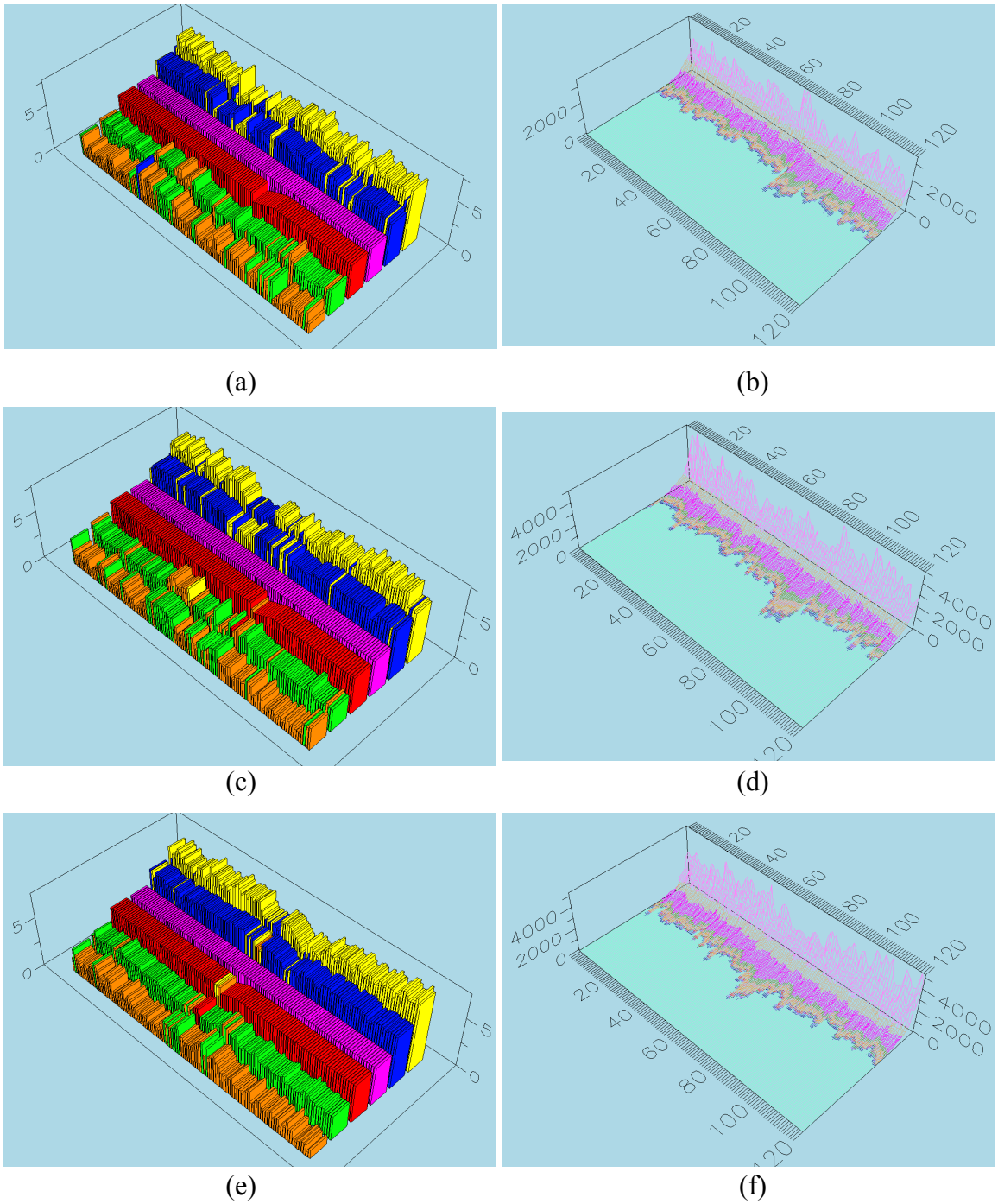


Figure 17 Screen shots of the low rate SL scale experiment results – (a) Thermal Towers with SL = 3, (b) Thermal Canyon with SL = 3, (c) Thermal Towers with SL = 4, (d) Thermal Canyon with SL = 4, (e) Thermal Towers with SL = 5, (f) Thermal Canyon with SL = 5

a. Conclusions

It appears that a less than linear relationship exists due to constant factors like transaction time. The graphs in Figure 17(c) and (d) look the best out of that set, resulting in an SL value of 4 seconds. The next experiment looks at varying the SR with a constant SL.

2. Constant SLIDELENGTH Ratio Experiment

This experiment was designed to explore the effects of changing the SR with a constant SL of 4 seconds. The values of SL, WL, SR, and the corresponding figures for each run of the experiment are outlined in Table 7.

WINDOWLENGTH	SLIDELENGTH	SR	Figures
16	4	4	Figure 18 a & b
24	4	6	Figure 18 c & d
32	4	8	Figure 18 e & f

Table 7 A table containing the configuration parameters for the low rate constant SL ratio experiment and the associated figures depicting the results.

It is clear that of the three sets of graphs in Figure 18, (e) and (f) are best configured for anomaly detection. This corresponds to a WL of 32 seconds resulting in an SR of 8.

a. Conclusions

Determining the SL, WL, and SR for traffic that is outside the optimal 2 to 3 kpps range is not a simple task. The scaling relationship is not a simple linear relationship. In fact, it is actually multidimensional, since the SL and the WL/SR were required to be changed by different factors. While the SL increased by a factor of 4, the WL was increased by a factor of 3.2. As a result, the SR was decreased by a factor of 0.8. Due to constants that do not scale with traffic rate, like transaction time, there is a dimensional scaling component. This dimensional scaling component increases the complexity of determining SL, WL, and SR for traffic levels outside the optimal range. As shown in Figure 18, it is possible to find an acceptable configuration for anomaly detection. This is accomplished by first optimizing the SL and then the WL.

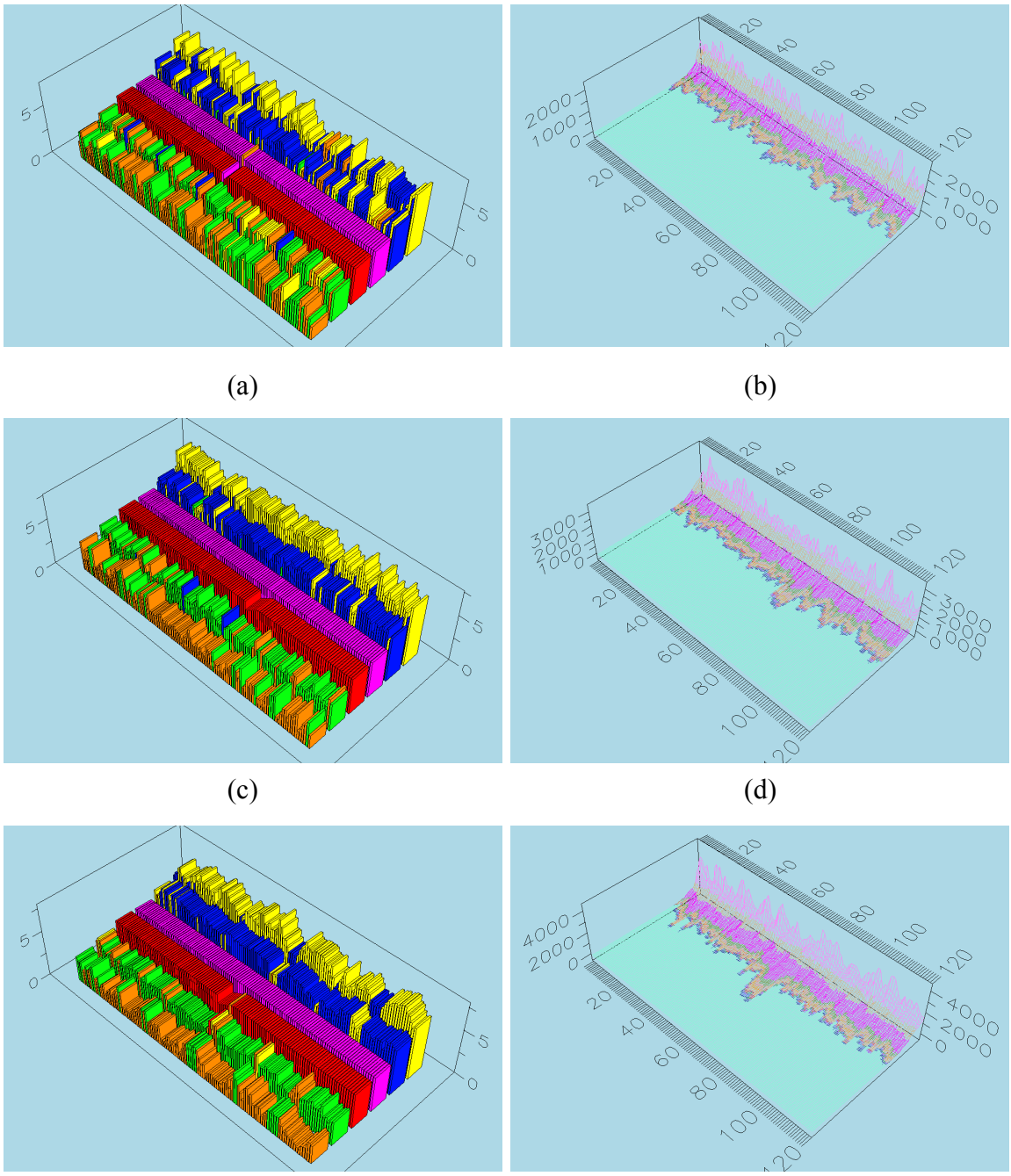


Figure 18 Screen shots of the low rate const SL ratio experiment results – (a) Thermal Towers with SR = 4, (b) Thermal Canyon with SR = 4, (c) Thermal Towers with SR = 6, (d) Thermal Canyon with SR = 6, (e) Thermal Towers with SR = 8, (f) Thermal Canyon with SR = 8

H. WHOLE BUCKET EXPERIMENT

All of the proceeding experiments were run with the same traffic configuration. This next section explores the effects of expanding the traffic configuration while keeping the Therminator configuration parameters constant. In the previous experiments, traffic flowed between two sets of bucket pairs, thus only using four of the six buckets for normal traffic. In this experiment, the system was tricked into thinking that some trusted clients were trusted servers and vice versa. This produced traffic that flowed between two bucket groups as shown in Figure 19.

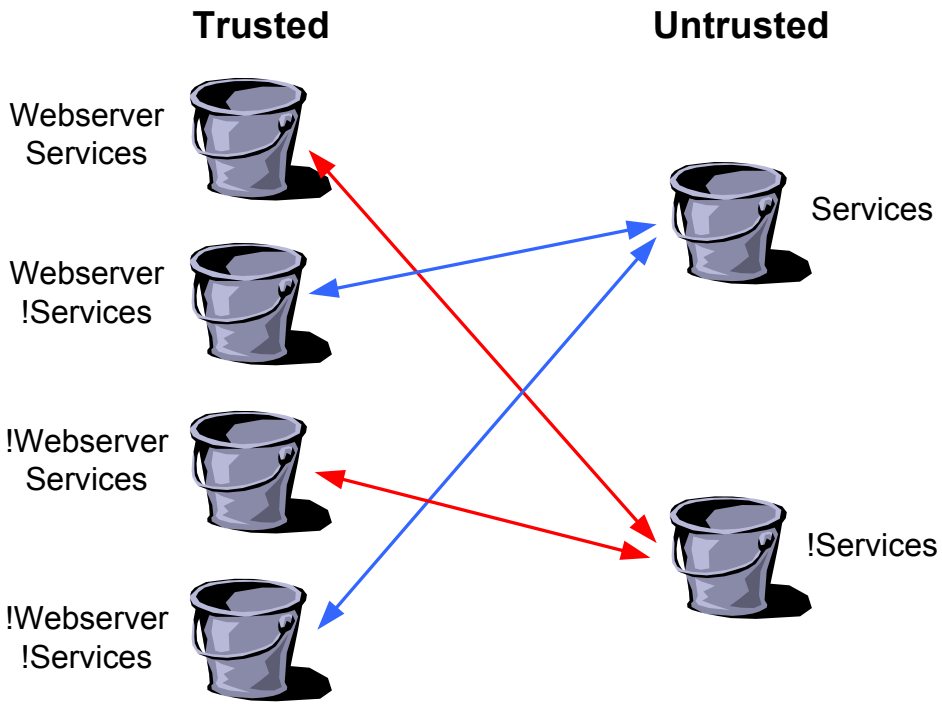


Figure 19 A graphic representing the two conversation triplets in the whole bucket experiment. Untrusted clients exchange data with untrusted servers and trusted clients exchange with untrusted servers.

This configuration utilized all six buckets for normal traffic. The traffic rate in the trusted buckets was approximately half that of the untrusted buckets. The settings and corresponding figures for the three runs of the experiment are shown in Table 8. All configuration files used for this experiment are contained in Appendix A.

WINDOWLENGTH	SLIDELENGTH	SR	Figures
10	1	10	Figure 20a
20	2	10	Figure 20b
30	2	15	Figure 20c
10	1	10	Figure 20d

Table 8 A table containing the configuration parameters for the whole bucket experiment and the associated figures depicting the results.

This experiment produced some very interesting results, shown in Figure 20(a) and (d). From the Thermal Towers, it can be seen that some buckets were properly smoothed, while others were very noisy. The change in configuration caused the trusted traffic to be split between two buckets in each pair. For example, in the previous experiments, traffic flowed between the blue and green buckets as well as the yellow and orange buckets. The new traffic configuration caused traffic to flow between the blue, green, and magenta buckets as well as the yellow, orange, and red buckets. The traffic that was going exclusively to the blue bucket is now split between the blue and magenta buckets. This also holds for the yellow and red buckets. This creates more than 100 unique states, as seen in the overflow of the Thermal Canyon graph. The Thermal Canyon graph did not change significantly in the other runs of the experiment, so it is only shown for the first experiment.

The experiment was then repeated, increasing the SL and decreasing the WL (to keep the SR constant). The results can be seen on Figure 20(b). Increasing the SL caused the yellow and orange buckets, the two with the most traffic, to become overly smoothed. The noise was slightly reduced in the other buckets, but overall there was little improvement. Next, the original SL was used, but the WL was increased to achieve a SR of 15. The results of this experiment can be seen in Figure 20(c). Again, the green and orange buckets are overly smoothed, but the noise is significantly reduced in the rest.

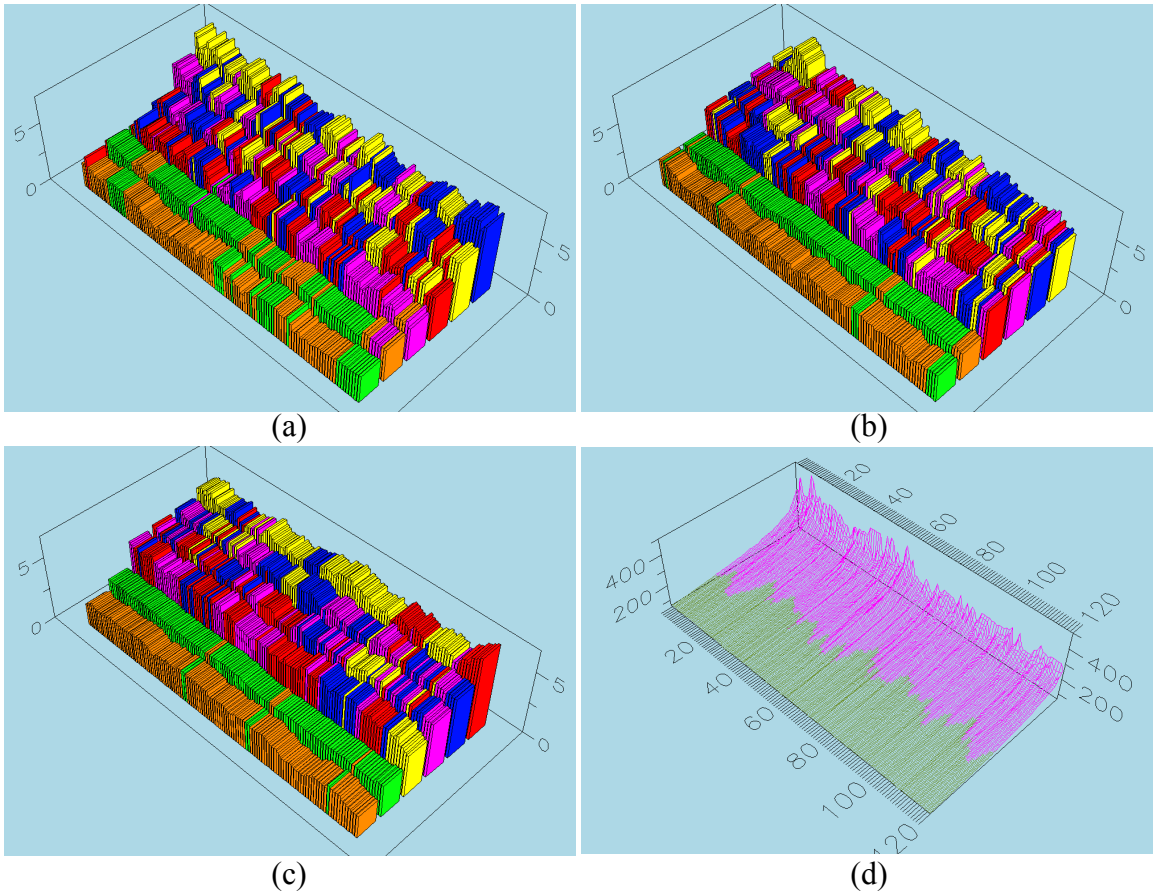


Figure 20 Screen shots of the whole bucket experiment results – (a) Thermal Towers with SR = 10, (b) Thermal Towers with SR = 10, (c) Thermal Towers with SR = 15, (d) Representative Thermal Canyon

1. Conclusions

From the results of the whole bucket experiment, it can be concluded that the proper SL setting depends on the distribution of traffic between the buckets. For configurations that have effectively equal traffic distribution between buckets, the SL and WL should be set as specified in the previous sections. When the distribution of traffic is not close to being equal, then the SL and WL need to be set according to the highest priority buckets. For example, in the previous experiment, the SL and WL should be set as in Figure 20(c). Since the information in the trusted buckets is of more concern than that of the untrusted buckets. Chapter VI explores changing the bucket boundaries in order to overcome unequal traffic distribution between buckets.

I. SUMMARY

The performance of Therminator is highly dependent upon proper configuration. The concept of a smoothing ratio, defined in Equation (6.1), has a filtering effect on the Therminator graphs. It was shown in this chapter that for traffic levels within the range of 2 to 3 kpps an SL of one and an WL of ten (SR = 10) is ideal (given boundary conditions listed in Table 3). The effects of changing SL and the WL were also shown for traffic rates within this ideal range.

Given that it is not always possible to ensure that the network traffic is within the ideal range, the scaling of these values for lower traffic rates was explored. It was shown that the values do not scale linearly, but more likely involve some sort of dimensional scaling function. Proper setting of the SL and WL could be determined by trial and error for lower traffic rates. In this case, the output is first optimized by setting the SL and then further optimized by adjusting the WL. For traffic rates lower than the optimal range, the SL must be increased as well as the WL, but the resulting SR will be lower.

Last, it was shown that these setting are based on the assumption that traffic is evenly distributed between buckets. It was shown that when there are gross differences in the traffic distribution between buckets, then the graph cannot be optimized for every bucket. In some cases this may be acceptable. In Chapter VI, the BUCKETSPACEINIT configuration parameter set and optimizing a system with an unequal distribution of traffic between buckets is explored.

THIS PAGE INTENTIONALLY LEFT BLANK

VI. CONFIGURATION – BUCKETSPACEINIT

A. CHAPTER OVERVIEW

In this chapter, the initial and boundary conditions for a bucket set are defined and explored. This consists of several experiments that examine the effects of changing the initial and boundary conditions on the Therminator GUI. The last section looks at optimizing a system with unequal traffic distribution between buckets.

B. INITIAL AND BOUNDARY CONDITIONS

The BUCKETSPACEINIT parameter set consists of the initial and boundary conditions for the buckets within a given bucket space. An example of a BUCKETSPACEINIT parameter set can be seen in Figure 21.

```
BUCKETSPACEINIT SIXBUCKET
  (0): 0/10 : 4
  (1): 0/10 : 4
  (2): 0/10 : 4
  (3): 0/10 : 4
  (4): 0/10 : 4
  (5): 0/10 : 4
ENDBUCKETINIT
```

Figure 21 An example of the BUCKETSPACEININT Parameter set in the <name>.config file for the Therminator PID5 executable. Each numbered line defines the lower/upper boundary limits and the initial number of balls for a bucket.

In the example above, the BUCKETSPACEINIT parameter set is named SIXBUCKET. It consists of six buckets, numbered from zero to five. Each bucket has a set of boundary conditions and an initial condition. In the example above, each bucket has the same boundary and initial conditions. The first two numbers after the bucket number are the boundary conditions. The boundary conditions consist of two limits. The first is a lower boundary limit, or the minimum number of balls a bucket may contain. The second is an upper limit, or the maximum number of balls a bucket may contain. In this example, each bucket may contain a minimum of zero balls and a maximum of ten balls. The next value is the initial condition, the initial number of balls the bucket contains. For this example, each bucket initially contains four balls. The BUCKETSPACEINIT parameter set

can be found in the <name>.config file for the Therminator PID5 executable. Appendix A contains an example of this file.

Exploring the boundary conditions is not as straight forward as it may seem. Besides the actual values of the boundary conditions, the relative distances between each of the boundary conditions are important. There are three relative distances of importance. The first is the distance between the lower boundary and the initial condition. The second is the distance between the initial condition and the upper boundary. The last is a function of the first two, the distance between the lower and upper boundary or the sum of the first two distances. The initial and boundary conditions affect the size of the bucket state space. The size (number of possible bucket states) is an important metric in optimizing Therminator performance.

C. INITIAL CONDITION EXPERIMENT

This experiment was designed to explore the effects of changing the initial condition, the distances between the initial ball count and the minimum and maximum ball limits. The distance between the boundary conditions remained constant, but the size of the bucket state space changes. The same test traffic that was used in Chapter V was used for this experiment. The configuration parameters used in this experiment are defined in Table 9.

SLIDELENGTH	WINDOWLENGTH	Smoothing Ratio	Boundary Conditions	Traffic Rate
1 second	10 seconds	10	0:10	2.4 kpps

Table 9 A table containing the configuration parameters for the initial condition experiment.

The initial condition was varied from 3 to 7 homogenously across the bucket space. The size of the bucket state space and the corresponding figures for the results of each run of the experiment are contained in Table 10. The symmetry of the bucket state space around the midpoint of the boundary condition (initial ball count equal to 5) is of significant interest.

Initial Ball Count	Total States*	Figures
3	49	Figure 22 a & b
4	81	Figure 22 c & d
5	121	Figure 22 e & f
6	81	Figure 23 a & b
7	49	Figure 23 c & d
* Not counting states added by anomalous packet		

Table 10 A table containing the initial ball counts for the initial conditions experiment and the associated figures depicting the results.

A small value for the initial condition results in a very small bucket state space. Each possible bucket state is frequently visited during each display period. This causes the Thermal Towers to become overly smooth, thus loosing information, as seen in Figure 22(a). The Thermal Canyon graph has very little variation due to the limited number of possible bucket states, as seen in Figure 22(b).

For an initial condition of 4, the graphs are the baseline configuration as explored in the previous chapter. Conversations can been seen in the Thermal Towers, Figure 22(c), and majority of the possible bucket states are being visited each display period, Figure 22(d), but not enough to saturate the graph. These figures were put in for comparison purposes.

An initial condition of 5 results in the maximum-size bucket state space. It is half way between the lower limit of 0 and the upper limit of 10. The conversations in the Thermal Towers are still apparent, as seen in Figure 22(e), although a subtle change occurs towards a rougher appearance. The change in the Thermal Canyon is the most noticeable. There is significantly more variability in the graph, Figure 22(f), as the system is allowed more possible bucket states. This produces a graph that is more difficult to interpret, since some normal activity appears anomalous.

In this configuration, there is symmetry about the midpoint initial condition of 5. Thus, the results for an initial condition of 6 are similar to those with an initial condition of 4. This holds true for initial conditions of 3 and 7. These results can be seen in Figure 23.

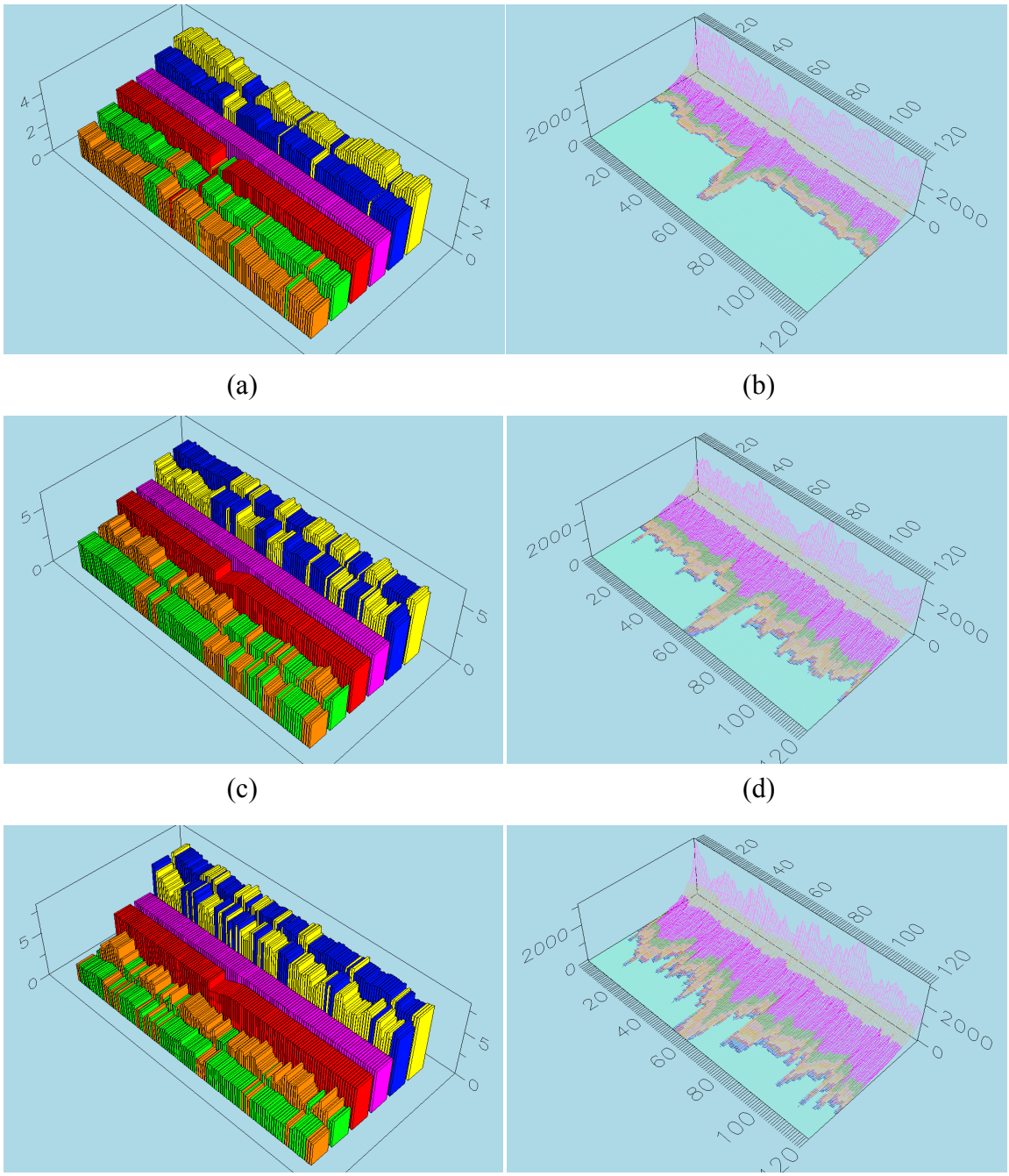


Figure 22 Screen shots of the initial conditions experiment results – (a) Thermal Towers with IC = 3, (b) Thermal Canyon with IC = 3, (c) Thermal Towers with IC = 4, (d) Thermal Canyon with IC = 4, (e) Thermal Towers with IC = 5, (d) Thermal Canyon with IC = 5.

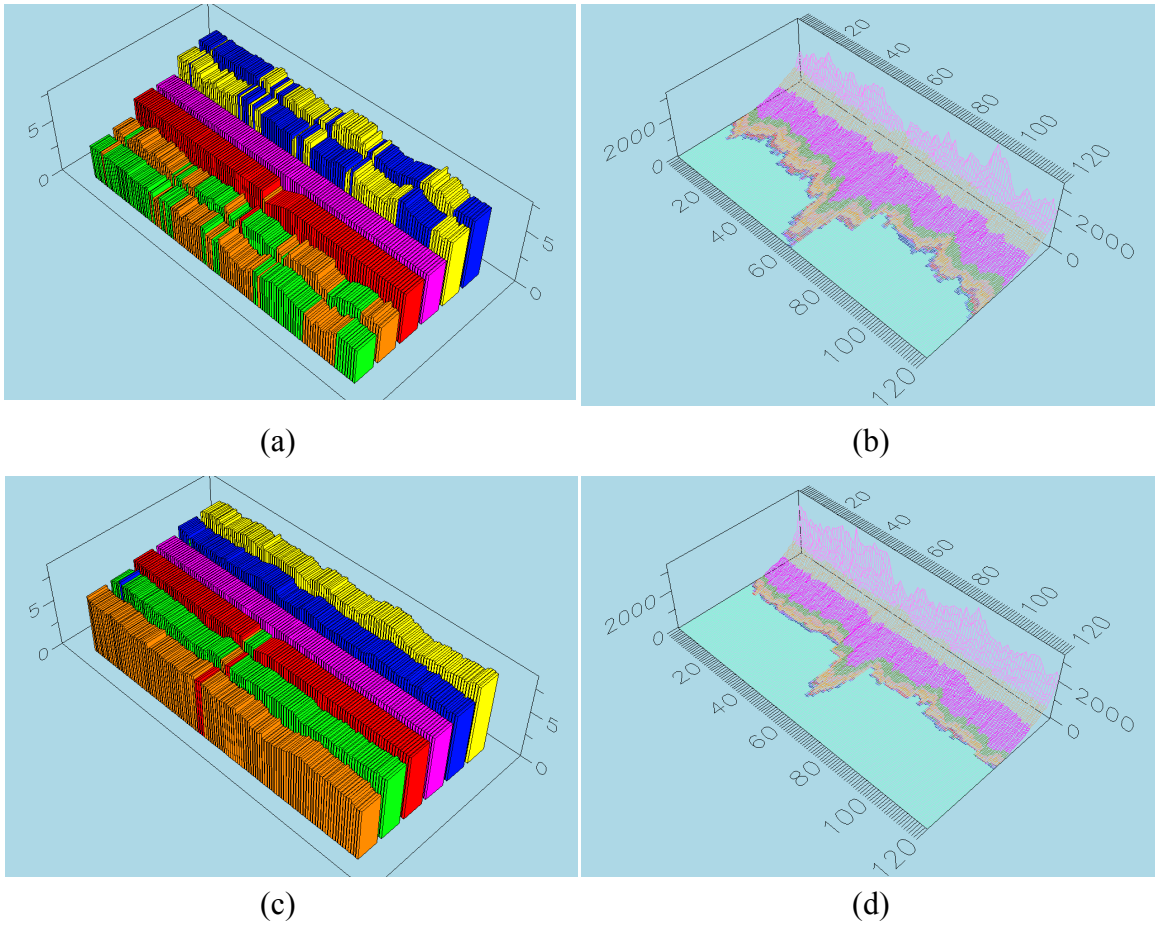


Figure 23 Screen shots of the initial conditions experiment results – (a) Thermal Towers with IC = 6, (b) Thermal Canyon with IC = 6, (c) Thermal Towers with IC = 7, (d) Thermal Canyon with IC = 7.

1. Conclusions

The size of the bucket state space is a major factor in the performance of Terminator. Changing the initial conditions has a dramatic effect on the bucket state space size. There is a symmetrical relationship for the initial conditions around the midpoint between the upper and lower boundary conditions. Reducing the initial condition reduces the size of the bucket state space. This has a similar effect as an increasing the traffic rate. Conversely, increasing the initial condition creates a larger bucket state space and has a similar effect as decreasing the traffic rate. These two rules are true for conditions below the point of symmetry and have the opposite effect above the symmetry point. Homogenous changes in the initial conditions are an effective way to tune system performance. Non-homogenous changes within a conversation group do not make sense unless a total ball

count other than $n*i$ is desired. The initial conditions should be kept homogenous between all buckets of a given conversation group. Bucket groups that are not involved in the same conversations may have different initial conditions.

D. BOUNDARY CONDITIONS EXPERIMENT

This experiment was designed to explore the effects of changing the boundary conditions on the Therminator GUI. Two aspects of boundary conditions were explored, homogenous and non-homogenous boundary conditions across the bucket space. In the first experiment, all of the boundary conditions are changed equally across the bucket space; n the second, only one bucket's boundary condition was changed. As in the previous section, changes in the boundary conditions affect the size of the bucket state space. The configuration parameters defined in Table 11 were used for each run of the experiment.

SLIDELENGTH	WINDOWLENGTH	Smoothing Ratio	Initial Condition	Traffic Rate
1 second	10 seconds	10	4	2.4 kpps

Table 11 A table containing the configuration parameters for the boundary conditions experiment.

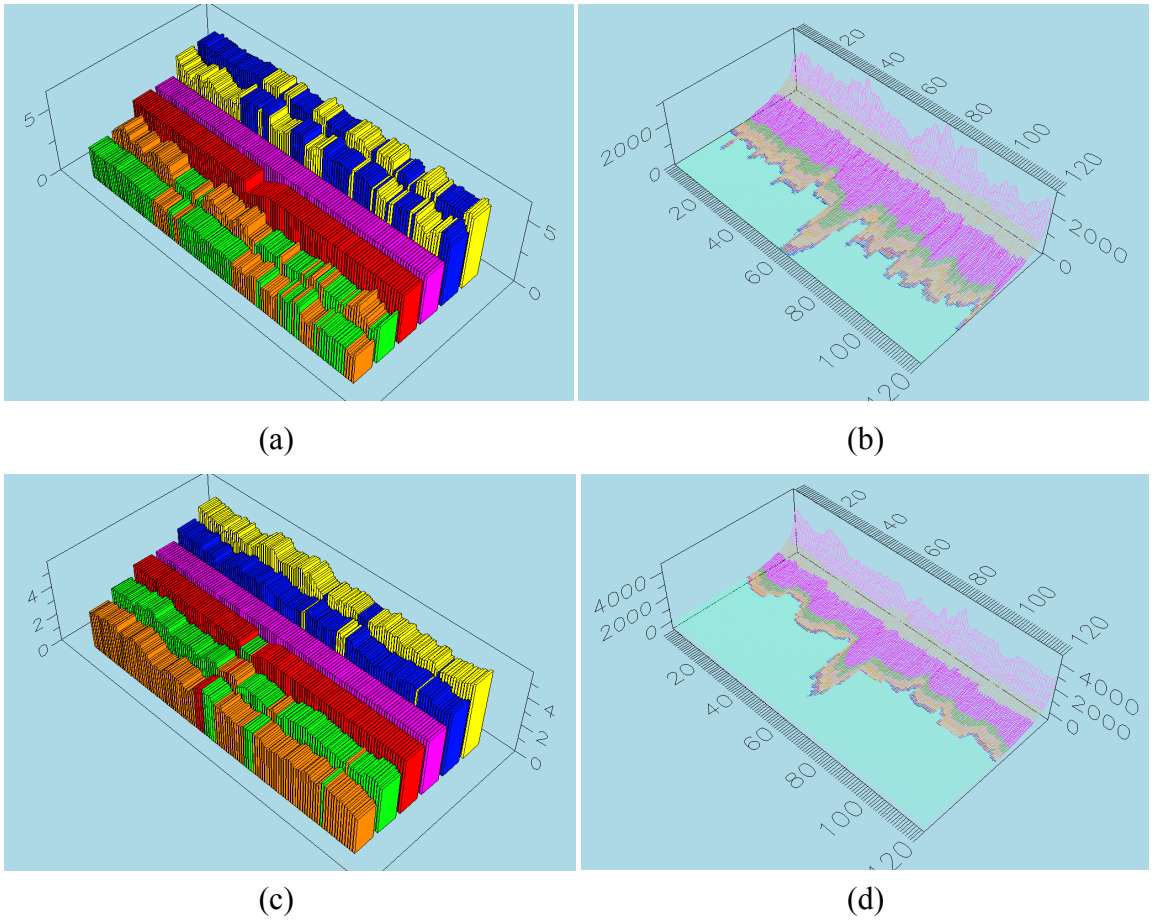
1. Homogenous Boundary Conditions

This experiment consisted of examining the effects of decreasing the distance between the boundary conditions homogeneously across the bucket space. For the ease of comparison, a base case is included with each run of the experiment. The initial and boundary conditions, size of the bucket state space, and corresponding figures for the results of each run are contained in Table 12.

Lower Limit	Upper limit	Initial Ball Count	Total States*	Figures
0	10	4	81	Figure 24 a & b
1	9	4	49	Figure 24 c & d
0	10	5	121	Figure 25 a & b
1	9	5	81	Figure 25 c & d
* Not counting states added by anomalous packet				

Table 12 A table containing the values of the boundary conditions for the homogenous boundary conditions experiment and the associated figures depicting the results.

The results displayed in Figure 24 (a) and (b) is used as a baseline for comparison to the results in Figure 24 (c) and (d). The distance between the boundary conditions was decreased by 2 in Figure 24 (c) and (d), the resultant graph becoming smoother. This was a similar effect to an increase in the traffic rate or decreasing the initial condition (assuming below the symmetry point).



(a) (b)

(c) (d)

Figure 24 Screen shots of the homogenous boundary conditions experiment results – (a) Thermal Towers with BC = 0/10 and IC = 4, (b) Thermal Canyon with BC = 0/10 and IC = 4, (c) Thermal Towers with BC = 1/9 and IC = 4, (d) Thermal Canyon with BC = 1/9 and IC = 4.

In fact, the configuration in Figure 24 (c) and (d) is identical to that of Figure 22 (a) and (b). There is a translation property associated with the boundary and initial conditions. Two BUCKETSPACEINIT parameter sets are equivalent if one set is a scalar translation of the other. For example the BUCKETSPACEINIT parameter sets listed in Table 13 are equivalent.

Lower Boundary	Upper Boundary	Initial Condition
0	10	4
5	15	9

Table 13 A table containing equivalent BUCKETSPACEINIT parameter sets. One set is a scalar transform of the other.

Repeating the experiment with another set of conditions, Figure 25 (a) and (b) is used as a baseline for comparison to the results in Figure 25 (c) and (d). The distance between the boundary conditions was again decreased by 2 in Figure 25 (c) and (d) in hopes of producing a smoother response. In fact, the configuration in Figure 25 (c) and (d) is identical to that of Figure 24 (a) and (b) based on the scalar translation property and the number of possible states as described below. Table 14 lists the BUCKETSPACE-INIT parameter sets for these sets of figures and the equivalency connection via the parameters in the middle row. No experiment was conducted for the case described in the middle row of Table 14, but the equivalency is explained below.

Lower Boundary	Upper Boundary	Initial Condition	Figures
0	10	4	Figure 24 a & b
0	8	4	N/A
1	9	5	Figure 25 c & d

Table 14 A table containing equivalent BUCKETSPACEINIT parameter sets. The first row is equivalent to the second since a two bucket conversation of 8 balls will never reach the upper boundary of 10. The third row is a scalar translation of the second row.

The parameter sets in the first two rows are equivalent. In this case, the conversation consists of two bucket pairs; therefore each conversation consists of 8 balls. With only 8 balls, any upper limit above the total number of balls will be equivalent. Rows two and three are then equivalent based on scalar translation. The difference between the parameter sets in the first two rows would only become evident if more balls are added to the conversation as from an anomalous event for example. In this case, the first row will produce a greater perturbation in the Thermal Canyon based on a larger bucket state space due to the added ball(s).

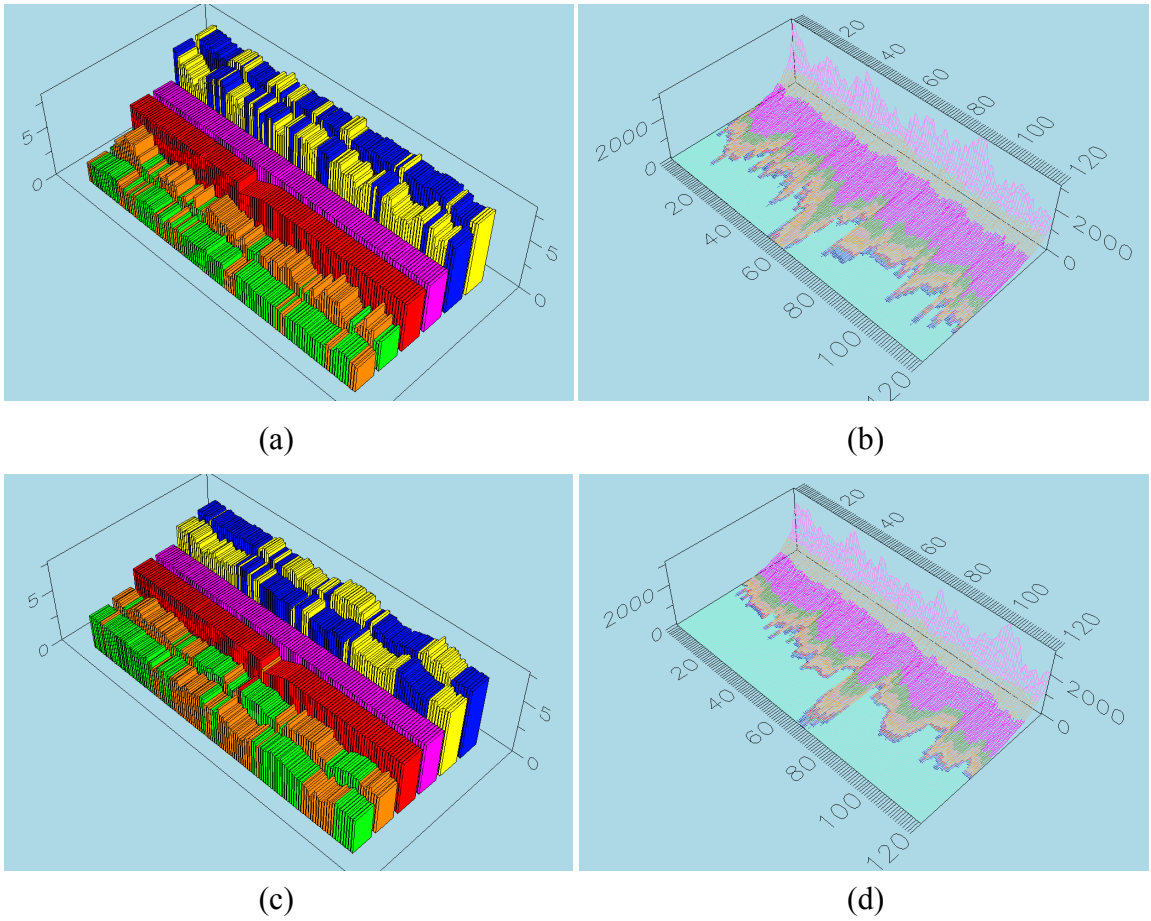


Figure 25 Screen shots of the homogenous boundary conditions experiment results – (a) Thermal Towers with BC = 0/10 and IC = 5, (b) Thermal Canyon with BC = 0/10 and IC = 5, (c) Thermal Towers with BC = 1/9 and IC = 5, (d) Thermal Canyon with BC = 1/9 and IC = 5.

2. Non-homogenous Boundary Conditions

This second experiment consisted of examining the effects of decreasing the distance between the boundary conditions for a single bucket. The baseline case is depicted in Figure 24 (a) and (b). The initial and boundary conditions and corresponding figures for the results of each run are contained in Table 15. The first run consisted of reducing the boundary condition in both directions. In the second run, only the upper boundary condition was reduced.

Run #	Bucket Number(s)	Lower Limit	Upper limit	Initial Ball Count	Figures
1	0	1	7	4	Figure 27a
	1 – 5	0	10		
2	0	0	6	4	Figure 27b
	1 – 5	0	10		

Table 15 A table containing the values of the boundary conditions for the non-homogenous boundary conditions experiment and the associated figures depicting the results.

In both cases, the results, depicted in Figure 26 were interestingly similar. Both runs had the expected affect on the bucket with the reduced boundary conditions as seen in the previous section. The unexpected result was the other bucket in the conversation being affected. The blue bucket had the reduced boundary condition, but the green bucket was affected as well. The green bucket was not affected as much as the blue bucket, but the effect was still significant. This experiment shows that a non-homogenous change in boundary conditions affects all of the buckets in a given conversation group.

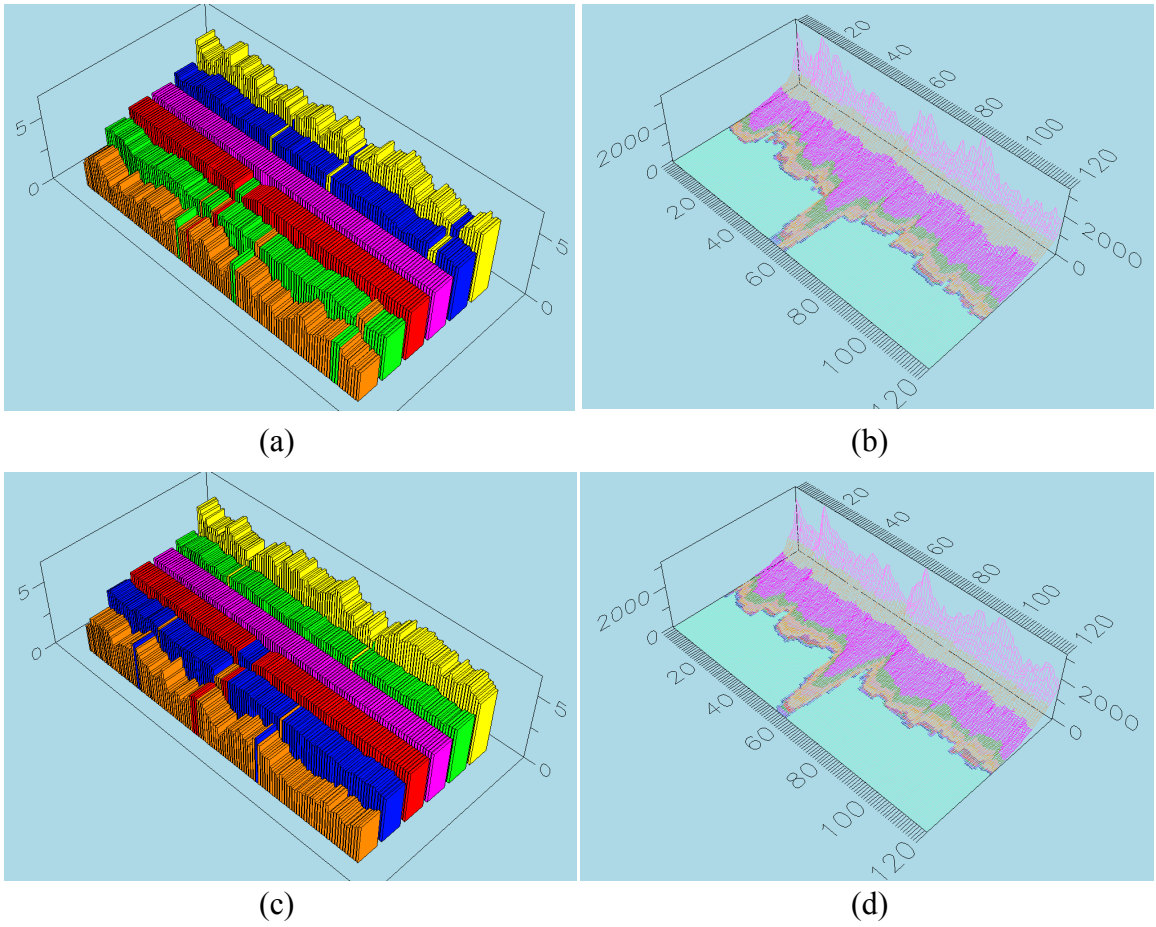


Figure 26 Screen shots of the non-homogenous boundary conditions experiment results – (a) Thermal Towers with $BC_0 = 1/7$, $BC_{1\text{-}5} = 0/10$, IC = 4, (b) Thermal Canyon with $BC_0 = 1/7$, $BC_{1\text{-}5} = 0/10$, IC = 4, (c) Thermal Towers with $BC_0 = 0/6$, $BC_{1\text{-}5} = 0/10$, IC = 4, (d) Thermal Canyon with $BC_0 = 0/6$, $BC_{1\text{-}5} = 0/10$, IC = 4.

3. Conclusions

Again, it is found that the size of the bucket state space is a major factor in the performance of Therminator. Reducing the distance between the boundary conditions reduces the size of the bucket state space. A reduction in the size of the bucket state space has a smoothing effect on the graph similar to increasing the traffic rate. Changing the boundary conditions can be the equivalent of changing the initial conditions due to the scalar translation property. Two BUCKETSPACEINIT parameter sets are equivalent if one is a scalar translation of another. It is therefore important to reduce the BUCKETSPACEINIT parameter set by the largest scalar possible without resulting in a negative lower boundary condition. This helps in understanding and ensuring the desired results

are obtained. Last, non-homogenous values for boundary conditions affect the entire conversation group; therefore, a conversation group is the smallest autonomous unit of a bucket space. The effect on the other buckets in the conversation group is not as pronounced, but still significant. Boundary conditions should be kept homogenous for all buckets involved in the same conversation group.

E. UNEQUAL TRAFFIC DISTRIBUTION EXPERIMENT

This experiment was designed to explore the possibility of optimizing the Terminator GUI for conditions in which traffic is unequally distributed among buckets within a conversation group. This was accomplished by adjusting the boundary conditions of individual buckets. The experiment parameters were those used in the whole bucket experiment in Chapter V.H. The key to optimizing this situation is in understanding the conversation flows. Figure 19 depicts the two conversation groups. The traffic rate in the trusted buckets is approximately half that of the untrusted buckets, but each conversation group contains both high and low traffic rate buckets. Therefore, in an attempt to optimize the system across the bucket space, the boundary conditions for the trusted buckets were reduced. The results of the non-homogenous boundary conditions experiment would imply that this will not achieve the desired results, but it was hoped that the effect on the lower traffic buckets would be more significant than the collateral effect on the higher traffic buckets. The experiment parameters defined in Table 16 where used for each run of the experiment.

Run #	Bucket Number(s)	SL	WL	SR	Lower Limit	Upper limit	Initial Ball Count	Figures
1	0 - 5	1	10	10	0	10	4	Figure 27a
2	0 - 3	1	10	10	1	9	5	Figure 27b
	4 - 5				0	12	5	
3	0 - 3	1	12	12	1	9	5	Figure 27c
	4 - 5				0	12	5	

Table 16 A table containing the values of the initial and boundary conditions for the unequal traffic distribution experiment and the associated figures depicting the results.

The first run establishes the base case for comparison, as seen in Figure 27(a). In the second run, optimization was attempted by reducing the boundary conditions for the lower rate buckets and increasing the boundaries for the higher rate buckets. The initial condition was also increased by one to provide a sufficient number of balls in the system. The results are depicted in Figure 27(b). The response in the Thermal Towers is still very rough in the lower rate buckets, but improved for the higher rate buckets. The third run improved the response by increasing the WL and thus the SR. This proved to produce a nice graph that would be effective in anomaly detection. In order to achieve this, all of the configuration parameters had to be modified. It is questionable whether the effort produces a configuration that is any better from that used in Chapter V.H that resulted in Figure 20(c). The question of how important the data in the high rate buckets are is vital in optimizing the system. In this case, the data is related to the untrusted network, and therefore of little value.

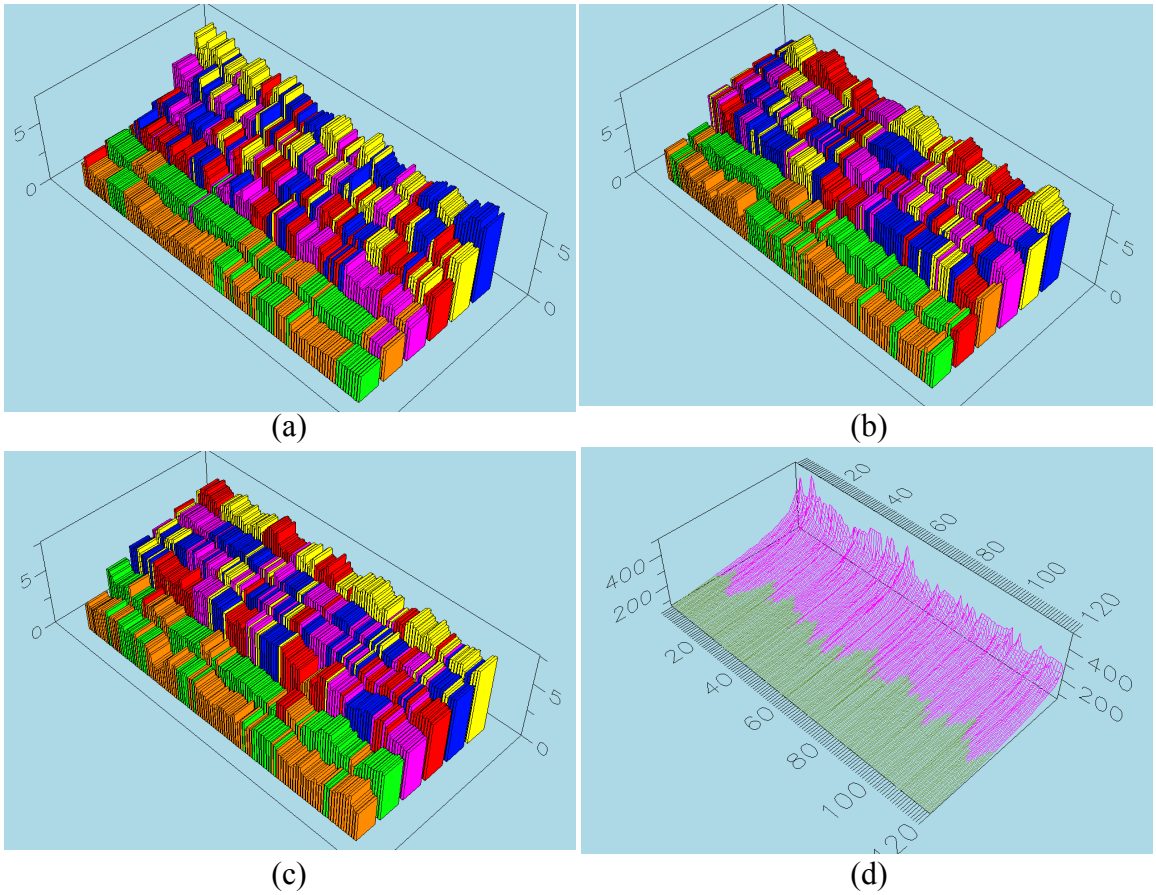


Figure 27 Screen shots of the unequal traffic distribution experiment results – (a) Thermal Towers with BC = 0/10, IC = 4, and SR = 10, (b) Thermal Towers with $BC_{0\text{-}3}$ = 1/9, $BC_{4\text{-}5}$ = 0/10, IC = 5, and SR = 10, (c) Thermal Towers with $BC_{0\text{-}3}$ = 1/9, $BC_{4\text{-}5}$ = 0/10, IC = 5, and SR = 12, (d) Representative Thermal Canyon.

1. Conclusions

With the smallest autonomous unit of a bucket space being a conversation group, the important relation becomes the size of the bucket state space within a conversation group. It is possible to improve the response of a system that has a non-homogenous traffic rate between buckets within the same conversation group. The process is difficult and requires manipulation of all the configuration parameters. It is questionable whether this provides any benefit over optimizing the configuration for the most important buckets.

F. SUMMARY

This chapter looked at the effects of altering the BUCKETSPACEINIT configuration parameter set. There were three sets of experiments conducted, exploring the initial conditions, boundary conditions, and the case of unequal traffic distribution across the bucket space. Any changes in the initial or boundary conditions affect the size of the bucket state space. The results of changing the size of the bucket state space were similar to those for changing the traffic rate. This implies a strong relationship between the size of the bucket state space and traffic rate. Last, it was determined that the smallest unit within a bucket space is a conversation group. Any changes to the initial or boundary conditions of a bucket affect all the buckets within the conversation group. It was found to be of little value to have non-homogenous initial or boundary conditions within a conversation group. The next chapter presents ideas to guide the creation of bucket spaces. These ideas were developed from observations made while conducting the experiments in Chapters V and VI.

THIS PAGE INTENTIONALLY LEFT BLANK

VII. CONFIGURATION – BUCKETSPACE

A. CHAPTER OVERVIEW

In this chapter, the BUCKETSPACE configuration parameters are defined. In addition, various ideas are expressed about configuring bucket spaces. Ideas formulated while performing and analyzing the results of the experiments in Chapters V and VI are presented. Last, the response of the system to anomalous packets is discussed.

B. INTRODUCTION

The BUCKETSPACE configuration parameters are what determine the decision tree and ultimately the bucket categories or bucket space. The settings for the BUCKETSPACE parameters can be found in the Therminator's executable <name>.config file. An example of this file can be found in Appendix A. The bucket space determines how the network participants are parsed and plays a critical role in optimizing the Therminator's performance. There settings are highly variable and depend largely on the traffic specifics of interest. For example, there can be multiple instances of Therminator running at one time, each displaying a different aspect of network traffic. One could be configured for the overall traffic on the network, one for DNS traffic, another for ICMP traffic, and lastly, one for web traffic. Each instance would require a different BUCKETSPACE configuration. The purpose of this chapter is to present some of the ideas that were discovered about BUCKETSPACE configuration while performing and analyzing the results of the experiments in Chapters V and VI. An in-depth look into BUCKETSPACE configuration is left for follow-on work.

C. TRAFFIC DISTRIBUTION

The expected traffic rate distribution between buckets should be equalized if possible and, at a minimum, within a conversation group. As seen in Chapter V.H and VI.E, when traffic is not equally distributed between buckets, the whole system becomes exponentially more difficult to configure. If the traffic cannot be parsed equally between buckets, then there are two possible solutions. The first solution is to assign a priority rating to each bucket based on the criticality of the information it contains. The system is then optimized for the highest priority buckets. The second solution is to tailor each

bucket's boundary conditions based on the expected level of traffic as explored in Chapter VI. This last method is only effective for unequal traffic distribution between conversation groups. Bucket prioritization is required for unequal traffic distribution within a conversation group.

D. DETECTING ANOMALIES

As stated in Chapter II, an anomaly can cause one of two effects in the Thermal Canyon graph. Either new states are visited, or previously visited states are seen more often. The first effect will cause a spike oriented along the z-axis and the latter along the y-axis. An anomaly that is orthogonal to the normal traffic flow will tend to cause a spike oriented along the z-axis, due to the new states visited. An anomaly that is parallel to the normal traffic flow will tend to cause a spike oriented along the y-axis, due to the revisiting of previously visited states. The magnitude of the potential spike is what determines the ability of the operator to detect the anomaly. The orientation of the anomaly with respect to the normal traffic flow will determine the magnitude of the perturbation. A single packet anomaly that is orthogonal to the normal traffic flow will cause a large perturbation in the graph, where a packet that is parallel to the normal traffic flow will cause a relatively small perturbation. The less orthogonal the anomaly is to the normal traffic flow, the larger the number of anomalous packets required to cause a noticeable perturbation in the graph.

For example, Figure 28(a) represents all of the possible bucket states that are contained in the bucket state space for a system consisting of three buckets (a, b, and c) each containing four balls. Each of the blue nodes represents a different bucket state. The number of balls in a given bucket is given by the lines parallel to the side opposite the vertex of interest. Each of the vertices corresponds to the case where all of the balls are in the associated bucket. The purple node represents the initial ball distribution, or initial bucket state of $\{4,4,4\}$. In Figure 28(b), the number of balls in bucket 'c' is constant at four. The thick green line represents the nine possible bucket states based on a conversation between the remaining two buckets.

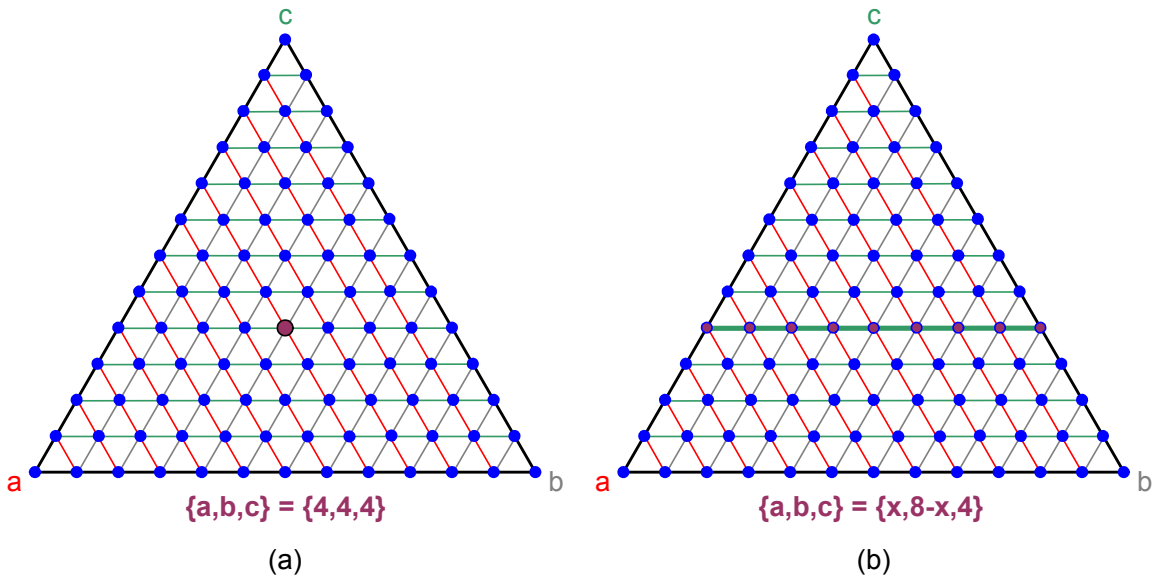


Figure 28 Graphics that depict the total bucket state space for a system containing three buckets each with four balls. Each node corresponds to a different bucket state. – (a) The purple node corresponds to the bucket state of $\{4,4,4\}$. (b) The green line represents the range of possible bucket states for a conversation between buckets 'a' and 'b'.

An example of the results of a single packet anomaly, that is orthogonal to the normal traffic flow, can be seen in Figure 29. In this case the packet caused a ball to move from bucket 'c' into the conversation between buckets 'a' and 'b'. The result is a new line of possible bucket states. This new line contains ten possible bucket states. Given that the data from any given SL is averaged over a WL of time, there are now nineteen possible bucket states, which is more than double the original number of nine. This results in a run out (in the z-axis direction) of the Thermal Canyon graph. This type of anomaly is very easy to detect even though it was caused by a single packet.

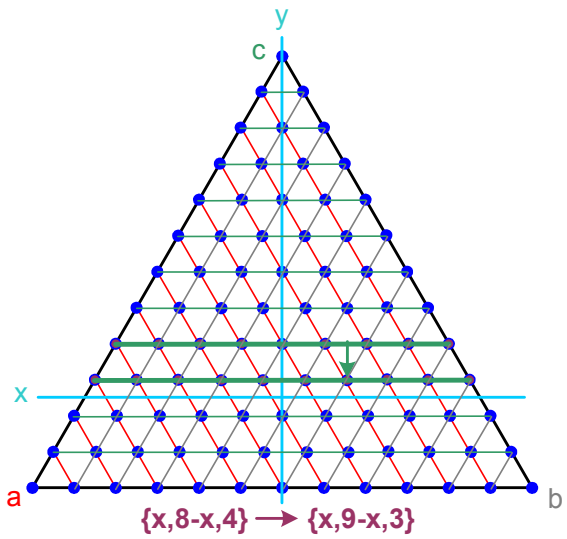


Figure 29 A graphic depicting the results of an anomalous packet that is orthogonal to the normal traffic flow. The anomalous packet causes a ball to move from bucket 'c' into the conversation between buckets 'a' and 'b'. The results is the state walk moves from the line at $c = 4$ to the line at $c = 3$.

Given that the Thermal Towers graph displays the average ball count per bucket per SL time period, it is less sensitive to anomalies that represent only a small percentage of the traffic. The Thermal Towers graph shows significant changes in the traffic flow. Therefore, if an anomaly is to be noticed in the Thermal Towers, it must comprise an appreciable percentage of the total traffic in the affected buckets. For example, a few packets that are parallel to the normal traffic flow of 500 pps will not be seen, but if the packets are orthogonal to the normal traffic flow, regardless of traffic rate, they will be seen.

The more orthogonal anomalous traffic is to the normal traffic flow, the greater effect the anomaly will have on the Therminator graphs. Since it is not possible to know all the expected anomalous traffic in advance, the key is to create a bucket space that provides tight classification of critical traffic. For example, traffic should be parsed by functional group, like web servers, as opposed to grouping servers and clients together. There is a limit to the number of buckets a configuration can have. Ideally, multiple instances of Therminator should be run concurrently to allow for smaller bucket spaces. This is also beneficial in reducing the complexity of interpreting the graphs which increases with the number of buckets.

E. SUMMARY

In this chapter, the BUCKETSPACE configuration parameters were defined as the categorization parameters that makeup a bucket space. Configurations resulting in unequal traffic distribution required either bucket prioritization or non-homogenous initial and boundary conditions. Last, anomalous traffic has one of two possible effects on the system, depending upon the orientation to the normal traffic flow. There is either an increase in the number of unique states visited or an increase in the number of times a given state is visited. The next chapter summarizes the findings from this study and presents areas for future work.

THIS PAGE INTENTIONALLY LEFT BLANK

VIII. SUMMARY AND FUTURE WORK

A. CHAPTER OVERVIEW

This chapter provides a summary of the findings of this study. Included in the summary are conclusions from observations made during the execution of this study. Suggestions for future and follow-on work are also presented.

B. SUMMARY

The first part of this study involved defining the bucket state space. Equations were developed for state counting to determine the size of bucket state space. These equations presented in Chapter III are necessary in understanding how to properly configure the Therminator. Next, the concept of a SR was developed to define the averaging time for data within the Therminator statistical mechanics algorithm. It was determined that a low SR results in increased false positives and a high ratio results in increased false negatives.

In exploring the SL, it was determined that the display period, which is equal to the SL, should be set based on the traffic rate. A relationship between the display period and traffic rate was not able to be empirically determined, but was found to be less than linear. It was also determined that optimizing a system for a reduced traffic rate would require a less than linear increase in the SL. The WL would also increase, but not as much as the SL. This results in a decrease in the SR.

It was determined that the SL and WL settings affect the bucket space as a system. Not all systems will have even traffic distribution across the bucket space. For situations with uneven traffic distributions, optimization can be achieved by changing the BUCKETSPACEINIT parameters. The smallest grouping within a bucket space was determined to be a conversation group. Uneven traffic distributions within a conversation group could not be optimized by changing the BUCKETSPACEINIT parameters. In this case, a bucket prioritization scheme must be used. For uneven traffic distribution between conversation groups, optimization could be achieved by changing the BUCKETSPACEINIT parameters. In this case, non-homogenous boundary conditions could optimize the system.

In experiments conducted on varying the BUCKETSPACEINIT parameters, it was determined that reducing the size of the bucket state space had a similar effect to increasing the traffic rate. In addition, the size of the bucket state space was symmetrical about the midpoint of the boundary conditions, assuming homogeneity across the bucket space with respect to the BUCKETSPACEINIT parameters. also, a translation property was discovered with the BUCKETPSACEINIT parameters. Two BUCKETSPACEINIT parameter sets were equivalent if one was a scalar translation of the other.

Lastly, the idea of orthogonal traffic was developed. The ability to detect anomalous traffic was determined to be based on the quantity of packets and their orientation to the normal traffic flow. The more anomalous packets received and/or the more orthogonal the traffic is to the normal traffic, the larger the perturbation in the Therminator GUI and thus a greater chance of being detected.

C. FUTURE WORK

This study explored two of the three major areas of Therminator configuration, the SL/WL and the BUCKETSPACEINIT. Ideas were presented for configuring the BUCKETSPACE, but no experiments were conducted. Properly configuring BUCKETSPACEs is a very complex task. The resultant bucket space is affected by the settings for the SL/WL and the BUCKETSPACEINIT. Further research could be conducted in the area of constructing bucket spaces.

This study showed that a relationship exists between the size of the bucket state space and the traffic rate. Further research could be conducted in determining the exact nature of this relationship. This examination may help to better quantify the relationship between the SLIDELENGTH and the traffic rate.

Most of this study was conducted using a simple bucket space and simple traffic consisting of HTTP exchanges. This was done to reduce the variability in conditions under which a basic understanding of the parameters could be gained. Further research could be conducted by looking at the response of more complex bucket spaces and traffic schemes to verify that the conclusions reached in this study hold for more complex situations.

APPENDIX A

The following is a representative <name>.config file for the Therminator PID5 executable used in all of the four bucket experiments.

```
# This is a standard Config file for the Thermal IDS Package
#
# NOTE: NO SINGLE LINE CAN BE GREATER THAN 256 CHARACTERS IN
  LENGTH
#
# <Gratuitous MetaPlot example>
# MASTERFILE FROM 2001.04.25.04:35:00+299 CHOOSE (SPORT[0|8]
  DPORT[0|8])
# CHOOSE (PROTO[ICMP]) CHOOSE (SIZE[28]) PLOT SIPvDIP EXPORT
  /tmp/b.out WITH TABS
# </Gratuitous MetaPlot example>
#

#   Counts/Lengths/Balls
InitialCount(4)
SLIDELENGTH(1.0)
WINDOWLENGTH(10.0)

#   Scale and Shift Signals on the Line Graph
TEMPSHIFT(0.0)
TEMPSCALE(10.0)
ENTROPYSHIFT(0.0)
ENTROPYSCALE(5000.0)
ENERGYSHIFT(0.0)
ENERGYSCALE(3.0)

# Begin Element Lists
IPLIST TRUSTED
  172.16.*.*
ENDLIST

IPLIST WEBSRVR
  172.16.10.110
  172.16.10.111
  172.16.10.112
  172.16.10.113
  172.16.10.114
  172.16.20.115
  172.16.20.116
  172.16.20.117
  172.16.20.118
  172.16.20.119
  172.16.20.120
ENDLIST

PORTLIST SERVICES
 <1023
ENDLIST
```

```
#  Bucket Space Initialization and Boundary Definitions

BUCKETSPACEINIT SIXBUCKET
   (0): 0/10 : 4
   (1): 0/10 : 4
   (2): 0/10 : 4
   (3): 0/10 : 4
   (4): 0/10 : 4
   (5): 0/10 : 4
ENDBUCKETINIT

BUCKETSPACE SB
    (TRUSTED)  ? ($1):($2)
    (WEBSRVR)  ? ($3):($4)
    (SERVICES) ? (4):(5)
    (SERVICES) ? (0):(1)

    (SERVICES) ? (2):(3)
ENDBUCKET

# Bucket Space Run Statements
RUN SB WITH SIXBUCKET
```

The following is a representative <name>.config file for the Therminator PID5 executable used in all of the six bucket experiments.

```
# This is a standard Config file for the Thermal IDS Package
#
# NOTE: NO SINGLE LINE CAN BE GREATER THAN 256 CHARACTERS IN
  LENGTH
#
# <Gratuitous MetaPlot example>
# MASTERFILE FROM 2001.04.25.04:35:00+299 CHOOSE (SPORT[0|8]
  DPORT[0|8])
# CHOOSE (PROTO[ICMP]) CHOOSE (SIZE[28]) PLOT SIPvDIP EXPORT
  /tmp/b.out WITH TABS
# </Gratuitous MetaPlot example>
#

#   Counts/Lengths/Balls
InitialCount(4)
SLIDELENGTH(1.0)
WINDOWLENGTH(10.0)

#   Scale and Shift Signals on the Line Graph
TEMPSHIFT(0.0)
TEMPSCALE(10.0)
ENTROPYSHIFT(0.0)
ENTROPYSCALE(5000.0)
ENERGYSHIFT(0.0)
ENERGYSCALE(3.0)

# Begin Element Lists
IPLIST TRUSTED
  172.16.*.*
ENDLIST
```

```
IPLIST WEBSRVR
  172.16.10.110
  172.16.10.113
  172.16.10.114
  172.16.20.117
  172.16.20.118
  172.16.20.120
  172.16.100.*
ENDLIST

PORTLIST SERVICES
 <1023
ENDLIST

#  Bucket Space Initialization and Boundary Definitions
BUCKETSPACEINIT SIXBUCKET
   (0): 1/10 : 4
   (1): 1/10 : 4
   (2): 1/10 : 4
   (3): 1/10 : 4
   (4): 0/10 : 4
   (5): 0/10 : 4
ENDBUCKETINIT

#  Bucket Space Definitions
BUCKETSPACE SB
    (TRUSTED) ? ($1):($2)
    (WEBSRVR) ? ($3):($4)
    (SERVICES) ? (4):(5)
    (SERVICES) ? (0):(1)
    (SERVICES) ? (2):(3)
ENDBUCKET

# Bucket Space Run Statements
RUN SB WITH SIXBUCKET
```

The following is the sucker.conf file for the Therminator PID5 executable used in all of the experiments.

```
#Sucker Configuration File
#

#Monitored Network
HOMENET 172.16.0.0/16

#Perform matching if set to 1; otherwise set to 0
MATCH 1

#Number of slides to keep in memory to perform matching
#  see SLIDE_LEN below.
MATCH_TIME 5

#Perform new friend functionality if set to 1; otherwise set to 0
FRIEND 0

#How many seconds the friend will remain in memory
```

```
FRIEND_TIME 3600
INT_FRIEND_TIME 1200

#How many seconds the friend bit will be set
FRIEND_DUR 0
INT_FRIEND_DUR 0


#How often (seconds) the sensor will send data to the RTDC
 SLIDE_LEN 1

#IP and Port of RTDC
RTDC_IP 127.0.0.1
RTDC_PORT 7777

FILTER ip
```

The following is the sensor.config file for the Therminator sucker executable used in all of the experiments.

```
# Sensor Configuration File
LISTEN_PORT 7777
KEYFILE sensor.pem
PASSWORD password
DHFILE dh1024.pem
CALIST calist
RANDOM random.pem

SENSOR_NAME box
SENSOR_IP 127.0.0.1
SENSOR_DESC tape

# Database specific
DATABASE PID
SENSOR_TABLE Sensor
THERM_TABLE Thermalate
```

LIST OF REFERENCES

[1] Thomas A. Longstaff, James T. Ellis, Shawn V. Hernan, Howard F. Lipson, Robert D. McMilliam, Linda Hutz Pesante, and Derek Simmel, "Security of the Internet," *The Froehlich/Kent Encyclopedia of Telecommunications Vol. 15,* Marcel Dekker, New York, pp. 231–255, 1997.

[2] Dorothy E. Denning, "An intrusion-detection model," *Proc. of IEEE Symposium on Security and Privacy*, pp. 118–131, 1986.

[3] M. Handley, V. Paxson, and C. Kreibich, "Network intrusion detection: Evasion, traffic normalization, and end-to-end protocol semantics," *Proc. of USENIX Security Symposium*, 2001, http://www.icir.org/vern/papers/norm-usenix-sec-01-html/, last accessed 3 December 2003.

[4] T. Ptacek and T. Newsham, "Insertion, evasion, and denial of service: Eluding network intrusion detection," Technical report, Secure Networks, Inc., January 1998.

[5] A. Rubin, *White-Hat Security Arsenal: Tackling the Threats*, Addison-Wesley, New York, 2001.

[6] D. Moore, C. Shannon, and J. Brown, "Code-red: A case study on the spread and victims of an internet worm," *Proc. of 2nd ACM Internet Management Workshop*, pp. 273–284, 2002.

[7] D. Moore, V. Paxson, S. Savage, C. Shannon, S Staniford, and N. Weaver, "The spread of the sapphire/slammer worm," Technical Report, CAIDA Technical Report, 2003

[8] "2003 CSI/FBI Computer Crime and Security Survey," Technical Report, Computer Security Institute, 2003, http://www.gocsi.com, last accessed 24 October 2003.

[9] D. Moore, C. Shannon, G. Voelker, and S. Savage, "Internet quarantine: Requirements for containing self-propagating code," *Proc. of IEEE INFOCOM*, pp. 1901–1910, March 2003.

[10] Raymond V. BonGiorni, Brian J. Eppinger, Peter L. Jobusch, and William E. Walker, "Thermonator: Analysis of the "Thermo" Patternless Intrusion Detection Research Prototype," Technical Report, National Security Agency, 16 Feb 2001.

[11] Donald, Stephen D. and McMillen, Robert V., "Therminator 2: Developing a Real Time Thermodynamic Based Patternless Intrusion Detection System," Masters Thesis, Naval Postgraduate School, Monterey, California, 2001.

[12] Stephen D. Donald, Robert V. McMillen, Dave K. Ford, and John C. McEachen, "Therminator 2: a thermodynamics-based method for real-time patternless intrusion detection," *Proc. of MILCOM 2002*, Volume: 2, pp. 1498–1502, Oct. 7–10, 2002.

[13] Ralph P. Grimaldi, *Discrete and Combinational Mathematics,* 4th Edition, Addison Wesley Longman, New York, 2000.

[14] SmartBits 6000B (SMB-6000B) Chassis Data Sheet*,* Spirent Communications Inc., Calabasas Hills, CA, 2003, http://www.spirentcom.com/documents/39.pdf, last accessed 24 October 2003.

[15] SmartBits LAN-3302A TeraMetrics Module Data Sheet*,* Spirent Communications Inc., Calabasas Hills, CA, 2003, http://www.spirentcom.com/documents/612.pdf, last accessed 24 October 2003.

[16] Avalanche SmartBits (SMB) Data Sheet, Spirent Communications Inc., Calabasas Hills, CA, 2003, http://www.spirentcom.com/documents/664.pdf, last accessed 24 October 2003.
[17] SmartBits SmartWindow Data Sheet, Spirent Communications Inc., Calabasas Hills, CA, 2003, http://www.spirentcom.com/documents/110.pdf, last accessed 24 October 2003.

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California

3. Chairman, Code EC
   Department of Electrical and Computer Engineering
   Naval Postgraduate School
   Monterey, California

4. Chairman, Code CS
   Department of Computer Science
   Naval Postgraduate School
   Monterey, California

5. Major General James Bryan
   Joint Task Force – Computer Network Operations
   Bethesda, Maryland

6. Admiral Richard Mayo
   Commander, Network Warfare Command
   Norfolk, Virginia

7. Admiral Joseph Burns
   Commander, Naval Security Group Command
   Fort Meade, Maryland

8. Ms. Rosemary Wenchel
   Chief Scientist, Naval Security Group Command
   Fort Meade, Maryland

9. Captain Michael Brown
   Commanding Officer
   Naval Information Warfare Activity
   Fort Meade, Maryland

10. Dr. William Semancik
    Director, Laboratory for Telecommunication Science
    Adelphi, Maryland